Introduction to IMS Version 8

This is the first of three presentations intended to provide an overview of the enhancements made available with IMS Version 8.

Enhancements addressed in these presentations fall generally into one or more of the following categories

- Availability and recovery
- Performance and capacity
- Application development
- Parallel sysplex exploitation
- Systems management

IMS

There are three presentations provided for the Technical Introduction to IMS V8. This first one describes the V8 enhancements that may be of interest to users of currently supported IMS function. The second addresses a new function introduced in IMS V8 that improves the systems management capabilities of IMS in a parallel sysplex.

A third presentation discusses IMS V8 installation and migration planning issues. It is not intended to be presented but is include as a supplement to the presentation material.

IMS V8 Contents

Availability and Recovery

- DBRC enhancements
  - 16MB RECON record size
  - PRILOG compression
  - RECON command authorization support
  - Automatic RECON loss notification
- Database Image Copy 2 enhancements
- ORS enhancements
- IMS/DB2 coordinated disaster recovery support
- Dynamic allocation of SLDS after IMS restart
- Batch Resource Recovery Services (RRS) support

IMS

During Part I of the presentation, we will talk about enhancements in each of these areas.

This visual shows those enhancements that improve the availability and recoverability of IMS systems and resources. Automatic RECON Loss Notification is described only briefly here and then expanded upon in Part II.

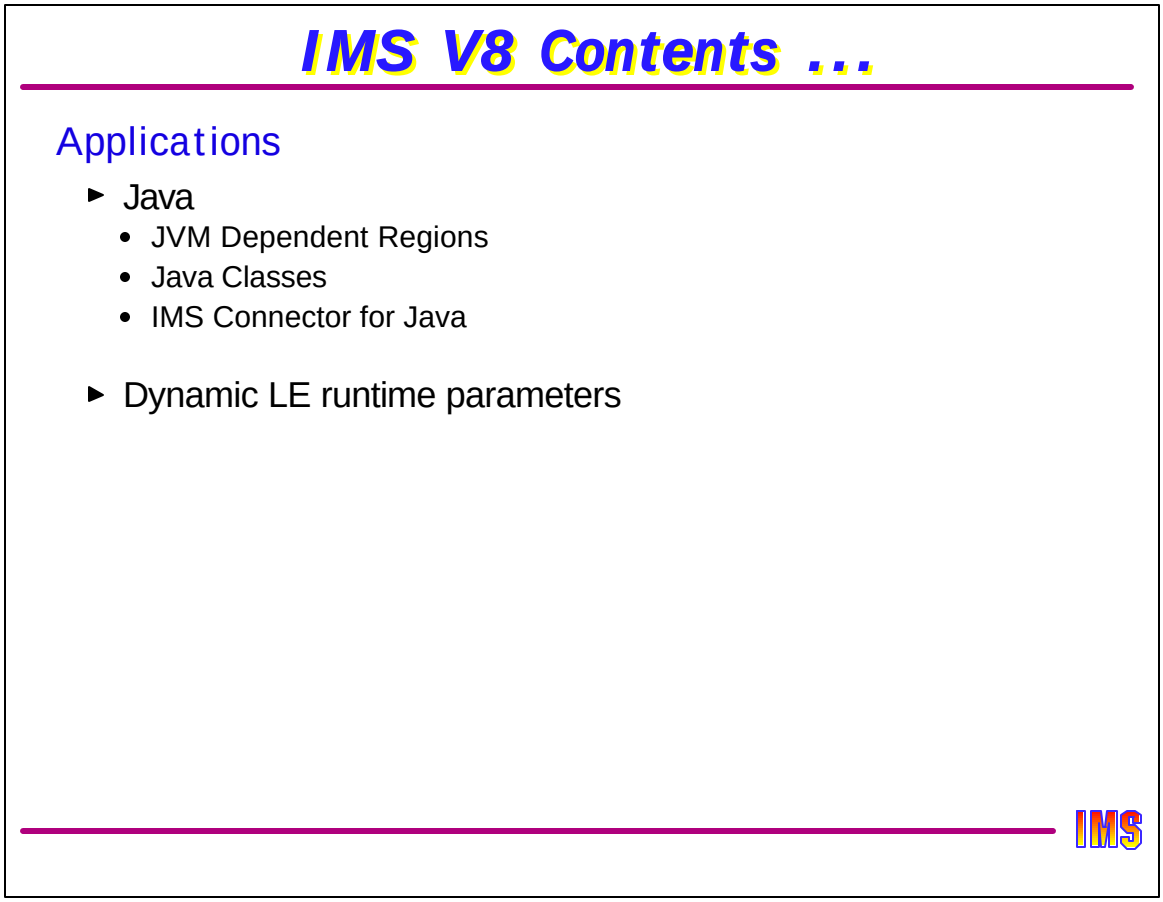
IMS V8 Contents ...

Performance and Capacity

- Parallel database open processing at restart
- Fast path DEDB enhancements
  - Support for 2048 areas
  - Support for Non-recoverable DEDBs
- VSCR for common areas (CSA and PVT)

IMS

There are few performance and capacity line items in V8, including an increase in the fast path DEDB maximum size to 2048 areas.



There has been a significant increase in the support of IMS for Java applications, including the introduction of Java Dependent Regions wich take advantage of the Persistent Reusable Java Virtual Machine environment for transaction processing.

Also, in Part II, we will describe how LE runtime options can be changed dynamically.

IMS V8 Contents ...

Systems

- APPC
  - Dynamic descriptors
  - Outbound LU support
  - CPU time limit for CPI-C transactions
- Syntax checker
- Dynamically allocate and read SLDS when needed after IMS restart
  - Available in V7 via SPE PQ50657
- Transaction trace

IMS

There are a few APPC enhancements as shown.

The Syntax Checker is an ISPF tool that allows you to edit and validate your DFSPBxxx proclib member. This is especially useful when migrating to a new version (e.g., V8) because it identifies new and obsolete parameters.

IMS V8 Contents ...

Parallel sysplex enhancements to existing function

- Coupling facility structure management
  - System managed duplexing
  - Systems managed rebuild
  - Automatic alter of structure size and entry-to-element ratio
  - Applies to
    - Fast path shared VSO structures
    - Shared queue structures
    - New CSL resource structure
- *Synchronous* APPC and OTMA shared queue support
  - IMS V7 provided support for *asynchronous* APPC/OTMA SQ support

IMS

There are some parallel sysplex enhancements offered for current users of data sharing or shared queues.

IMS V8 Contents ...

New and Improved IMS Architecture

- Common Service Layer (CSL)
- New address spaces
  - Structured Call Interface (SCI)
    - IMSplex member registration
    - Communications between IMSplex members
  - Operations Manager (OM)
    - IMSplex-wide command entry
  - Resource Manager (RM)
    - Global resource management
- New function supported in IMSplex
  - Sysplex Terminal Management (STM)
  - Single Point of Control (SPOC)
  - Coordinated Global Online Change (G-OLC)

IMS

Part II will address the new systems management capabilities when IMS is running in a parallel sysplex with the Common Service Layer - the next step in the evolution of IMS architecture.

Agenda

There are three parts to these presentations

- Part Ia - Base Enhancements
  - These are enhancements which may be applicable to all IMS V8 users, including those not using parallel sysplex functions in IMS
- Part Ib - Parallel Sysplex Enhancements
  - These are enhancements to IMS's current use of parallel sysplex functions
- Part II - Common Service Layer
  - This is new function in IMS V8 and represents an extension to the IMSplex architecture for systems management
- Part III - Installation and Migration Considerations
  - This part is for reference purposes only; it is not presented

IMS

The rest of this presentation will address everything except the Common Service Layer. Part II will be dedicated to CSL. Part III is not intended to be presented (unless time permits) but is included to identify some of installation and migration considerations for IMS V8.

The Base Highlights are those other than the Common Service Layer, which is presented in Part II.

Part Ia are non-sysplex enhancements.
Part Ib describes the parallel sysplex enhancements other than CSL.

IMS V8 Highlights

Enhancements to ...

- ★ Base IMS product without parallel sysplex
  - Database Manager
  - Transaction Manager
  - Systems
  - Applications

Enhancements to ...

- ★ Parallel Sysplex
- ★ Common Service Layer

IMS

This part of the presentation will address the base highlights associated with the IMS system components: DB Manager, Transaction Manger, Systems, and Applications.

Large RECON Record Size

IMS outages could be required because RECON records would exceed VSAM maximum record size

- PRILOG (family of) records
  - Entries for all archived logs
  - PRILOG compression not always effective
- DBDS record
  - DASD/CU problems may result in many I/O errors and EEQEs
- SUBSYS record
  - Large number of authorized databases
- LOGALL record
  - Large number of data sets with updates on log

Warning message DSP0278W - then U0071 if not corrected

- LOGALERT and SIZALERT provide warnings

IMS

One of the long-standing requirements against IMS was to remove the RECON record size as the cause for having to recycle IMS. If a RECON record is allowed to grow until it exceeds the VSAM defined maximum record size, then IMS will abend with a U0071. This is typically because the PRILOG (and related records) record got too large, but could possibly be because the DBDS, SUBSYS, or LOGALL record got too large.

Before the abend, LOGALERT and SIZALERT give a warning messages (DSP0287W) indicating that the size is becoming critical and that the operator should take action. Hopefully the operator can compress the PRILOG record, but sometimes the only action to take is to shut down IMS and restart it with a new PRILOG record.

16MB RECON Record Size

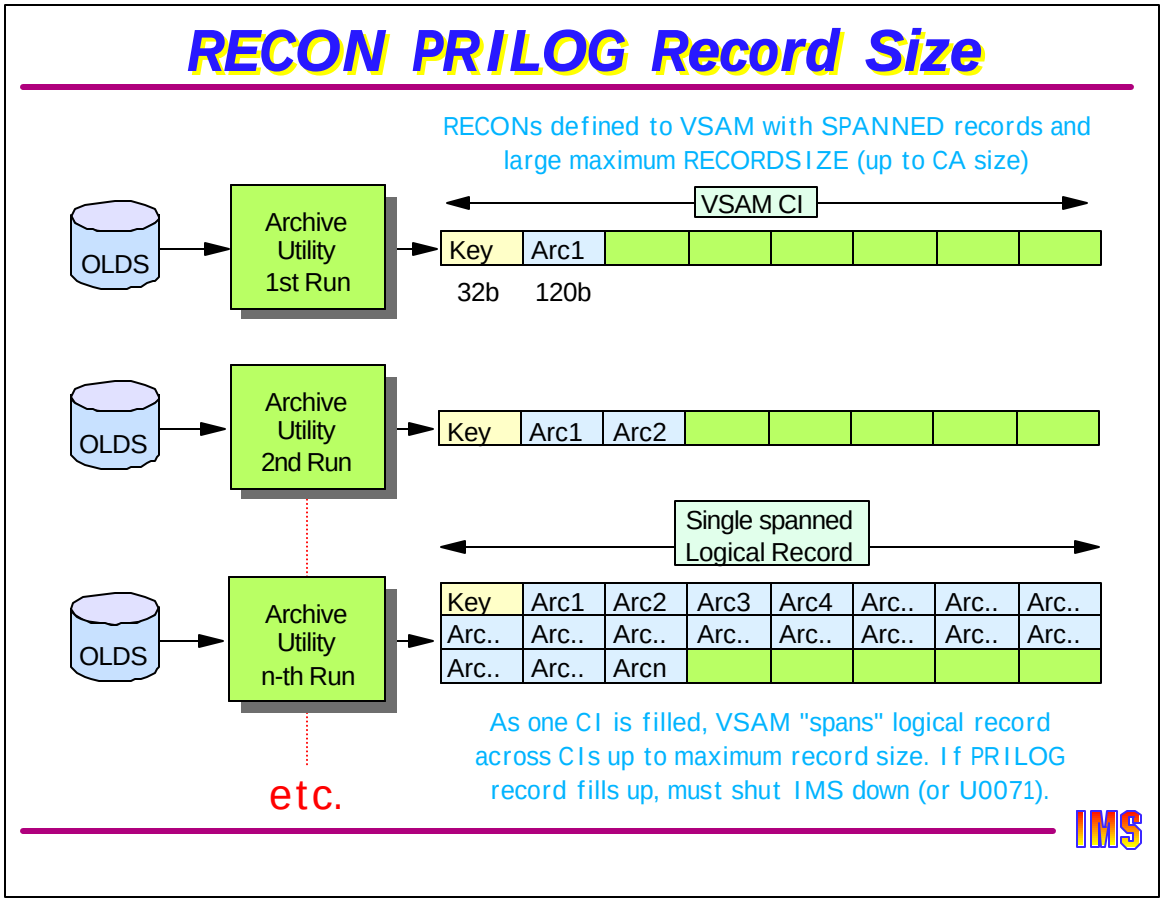
Large maximum record sizes are needed to extend IMS continuous availability

- Pre-V8 used VSAM *spanned* records
- Maximum record size limited to Control Area size
  - Typically one DASD cylinder (about .75 MB)
- Prevented RECON backup to tape if >32K
  - Had to back up to another KSDS

IMS

To get more than a 32K PRILOG record, the RECON data sets had to be defined as "spanned." When this was done, the maximum record size could be set anywhere up to a control area size, which is typically one DASD cylinder. For 3390 and 18K control intervals, this would be about 810K.

Spanning records to greater than 32K also meant that the RECONs could not be backed up to tape. Backups could only be done to another KSDS, making if difficult to ship the RECONs offsite.



Prior to IMS V8, the maximum size of the PRILOG record was determined by the maximum record size set by the VSAM DEFINE command. The maximum VSAM allows is when SPANNED records are defined, in which case the maximum can be up to the size of a control area (a DASD cylinder). If the size of the RECON record is greater than 32K, then it can't be backed up to tape - it could only be backed up to DASD.

Each time an OLDS is filled, an archive job is run to archive that OLDS to an SLDS and/or RLDS. The PRILOG (and PRISLDS) record is then updated with information about that archived OLDS. If the output SLDS is a single volume, then the archived information requires 120 bytes from the PRILOG record. Each additional archive takes another 120 bytes. When the VSAM CI is filled, the logical record is "spanned" to the next CI. This continues until either the PRILOG is compressed (/DELETE.LOG INACTIVE) OR IMS is shutdown. If neither is done, then the PRILOG will fill up and IMS will abend with a U0071. Prior to the abend, a warning message (DSP0287W) will be sent giving operations time (hopefully) to take corrective action.

16MB RECON Record Size

IMS V8 uses "segmented" records instead of VSAM spanned records

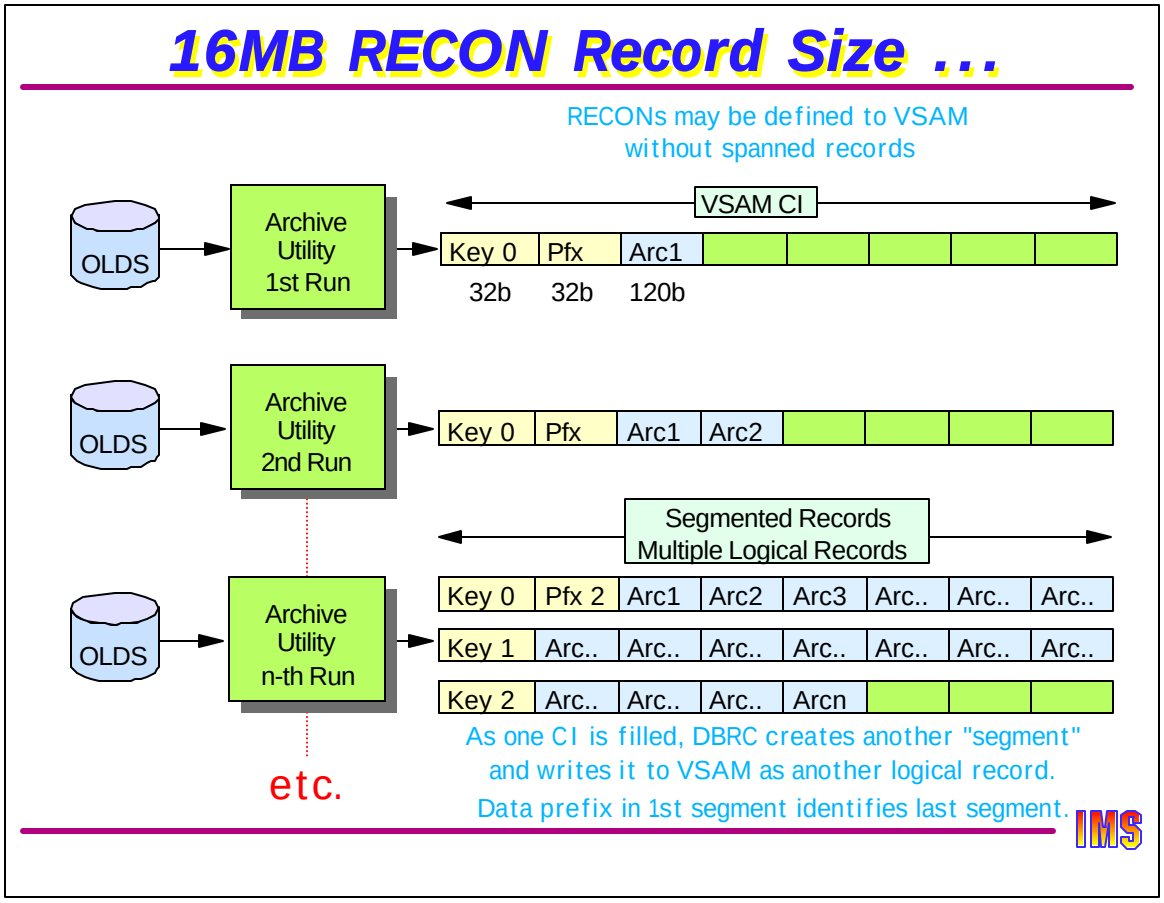
- Controlled by IMS
  - Not limited by VSAM maximum record size or control area size
- Records are divided into multiple segments - 1/CI
  - Last two bytes of KEY is Segment #
  - First segment contains DATA PREFIX with last Segment #
- Maximum total size of all segments - 16MB
  - *More than 100,000 single volume archived data sets*

IMS

IMS V8 does not use spanned records, even if the data set is defined with the spanned attribute and maximum record size greater than the CI size. Instead, IMS "segments" records into segments that are never larger than one CI, or one logical record, whichever is smaller. Each segment is a logical record. Since IMS will never write a logical record larger than a CI, VSAM will never attempt to "span" the record.

The last two bytes of the 32-byte RECON record key, unused prior to V8, is now used to indicate the segment number of a multi-segment RECON record. The first segment is segment number zero. The first segment has a 32-byte prefix following the key which indicates the segment number of the last segment in the RECON record. This tells DBRC how many segments are needed for a full RECON record just by reading the first segment.

In round numbers, and if every archive were a single volume SLDS, a 16MB PRILOG record could hold well over 100,000 SLDSs.



This diagram shows the first segment with a segment number "0" and a prefix indicating that "0" is the last segment in the record. When the first CI is filled, then a new VSAM logical record (segment "1") is written. The Prefix in segment "0" would be set to "1". This process continues until IMS is shut down and the PRILOG record is closed.

16MB RECON Record Size ...

Migration considerations

- IMS V8 supports RECONs with differing CI/record sizes
  - Can change one RECON at a time
  - Not recommended to be permanent
    - Requires additional overhead for segmenting
- RECON I/O Exit (DSPCEXT0)
  - Complete unsegmented record passed to exit
    - May be much larger than before (up to 16MB)
    - Length passed in standard parameter list
- *May want to adjust LOGALERT and SIZALERT*
  - Based on 16MB size - not VSAM maximum record size
- LIST.RECON displays current PRILOG record size
- SPEs available for V6 and V7
  - Will read and write segmented records, but ...
  - Still limited to maximum record size specified to VSAM
  - CI and logical record sizes must still be the same for all RECONs

IMS

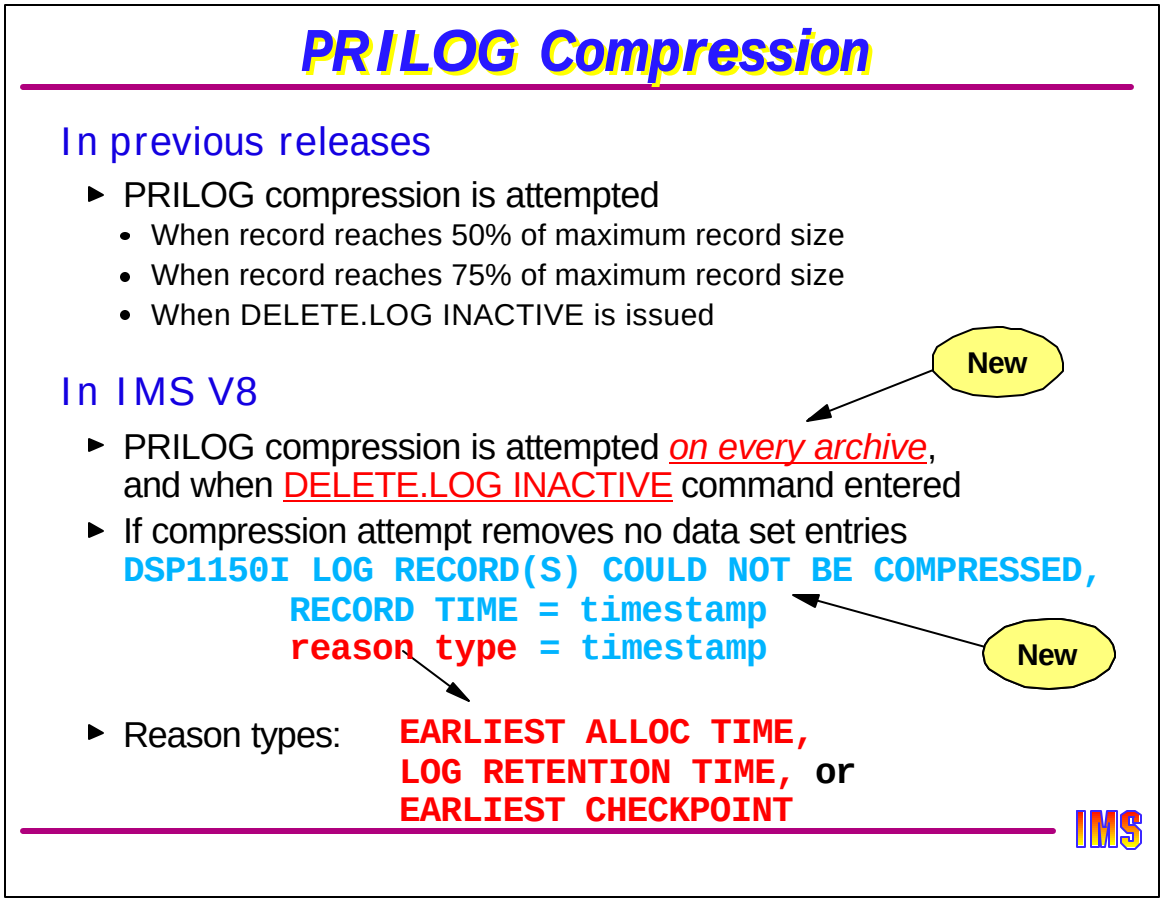
When running with IMS V8, you may want the CI sizes to be different than in prior versions. It is not necessary to change them all at once. If they are different sizes, then they can be changed one at a time online by changing the spare, then switching to the spare, then redefining a new spare, then switching again, and then the third. However, it is recommended that they all eventually have the same CI size and record size to avoid unnecessary segmenting overhead by segmenting different RECONs differently.

The RECON I/O Exit (DSPCEXT0) will not be presented with "segments." Instead, the exit will see, as in prior releases, a single large unsegmented record. The length of the total record will be passed in the Standard Parameter List. This exit should be evaluated prior to using it with V8 RECONs to be sure it can handle potentially much larger records.

LOGALERT and SIZALERT, introduced in IMS V7, are based on the amount of space left in the PRILOG record for archiving, and (in the case of SIZALERT) the percentage full of the maximum size. Since the maximum size is now 16MB, these parameters will have to be adjusted to make them reasonable. For example, you wouldn't want to wait until a 16MB record is almost full before issuing the DSP0287W warning. If the PRILOG record gets that large, there are likely to be other problems.

The LIST.RECON command output now displays the current size of the PRILOG record.

There are SPEs for IMS V6 and V7 that allow them to read and write segmented records, but they will NOT write more segments than would hold the amount of data set by the VSAM maximum record size. IN other words, if the maximum record size in V7 is 128K, then V7 DBRC will not write more than 128K worth of segmented data.



PRILOG compression eliminates data set entries from the PRILOG record. These data set entries contain information about archived logs that are no longer required for database recoveries or IMS restarts. Compression is attempted when archives are done and when the DELETE.LOG INACTIVE command is processed.

Previous releases of IMS attempted to compress the PRILOG when it grew to over 50% of its maximum record size. If the compression could not keep the record below 50%, compression was not attempted again until the record exceeded 75% of its maximum record size. Once it reached this level, compression was always attempted until it was reduced below 75%,

Since the maximum record size in IMS V8 is 16MB, the times at which compression is attempted is changed. It is now attempted on every archive.

If a compression attempt does not compress the record, a new information message (DSP1150I) is issued. It identifies the timestamp of the PRILOG, the reason that compression could not be done, and its timestamp. For example, if the oldest entry in the PRILOG is needed because an ALLOC record requires it, "EARLIEST ALLOC TIME" is included in the message along with that time.

For successful compressions, DSP0135I is issued as in previous releases.

IMS V8 keeps information about the oldest allocation for each DBDS in the LOGALL record. This reduces the overhead required for compression attempts.

LIST.LOG output includes the earliest overall ALLOC on the log and DBDSs sorted in order of their earliest ALLOC.

PRILOG Compression ...

Periodically, DBRC will attempt to remove unneeded logs from the PRILOG record, freeing up space.

Key 0	Pfx 2	Arc1	Arc2	Arc3	Arc..	Arc..	Arc..
Key 1	Arc..	Arc..	Arc..	Arc..	Arc..	Arc..	Arc..
Key 2	Arc..	Arc..	Arc..	Arcn			

DELETE.LOG INACTIVE

Key 0	Pfx 1	Arc7	Arc8	Arc9	Arc..	Arc..	Arc..
Key 1	Arc..	Arc..	Arc..	Arc..	Arcn		

Three conditions will prevent PRILOG compression

1. Log still needed for potential database recovery (oldest image copy older than log start time)
2. DBRC log retention time not expired
3. Log needed for /NRE BUILDQ (last SNAPQ or cold start on this log)

IMS

When PRILOG compression is successful, the SLDSs in the PRILOG record are "shifted" left. When this results in fewer segments being required, the eliminated segment is deleted from the RECONs (remember, these segments are VSAM logical records) and the prefix of the first segment is updated to reflect the new last segment.

DBRC Command Authorization

Security support for DBRC commands

- Commands can be authorized
  - At the *command verb* level
    - For example, the **CHANGE** command
  - At the *verb + resource type* level
    - For example, the **CHANGE.DB** command
  - At the *verb + resource type + resource name* level
    - For example, the **CHANGE.DB** DBD(**ACCTDB**) command
- Security profiles may differ for different RECONs
- Security is invoked only for commands issued from DBRC Utility (DSPURX00) or HALDB Partition Definition Utility
  - Use IMS command security for /RMx commands

IMS

IMS V8 introduces security support for DBRC commands when entered using the DBRC utility DSPURX00. The same security applies when using the HALDB Partition Definition Utility.

Commands may be secured at the command level. This secures all use of the command verb, such as a CHANGE command.

Commands may be secured at the command verb plus resource type level. This secures the command use with a type of resource, such as a CHANGE.DB command.

Commands may be secured at the command verb, resource type, and resource name level. An example of this is a CHANGE.DB DBD(AAA) command where AAA is a database.

Security may be implemented differently for different RECONs.

DBRC Command Authorization ...

Invoking Command Authorization

- Activate using DBRC command

CHANGE.RECON CMDAUTH(SAF|EXIT|BOTH|NONE,safhlq)

- SAF - invoke security product (e.g. RACF)
- EXIT - invoke DBRC Command Authorization exit routine (DSPDCAX0)
- BOTH - invoke both security product and exit routine
- NONE - do not invoke command authorization

- safhlq - RECON high level qualifier (e.g. IMSP)
  - To distinguish between different RECONs in RACF
  - 1 to 8 characters
  - Must be specified with SAF, EXIT, or BOTH
  - Cannot be specified with NONE

- LIST.RECON displays current settings

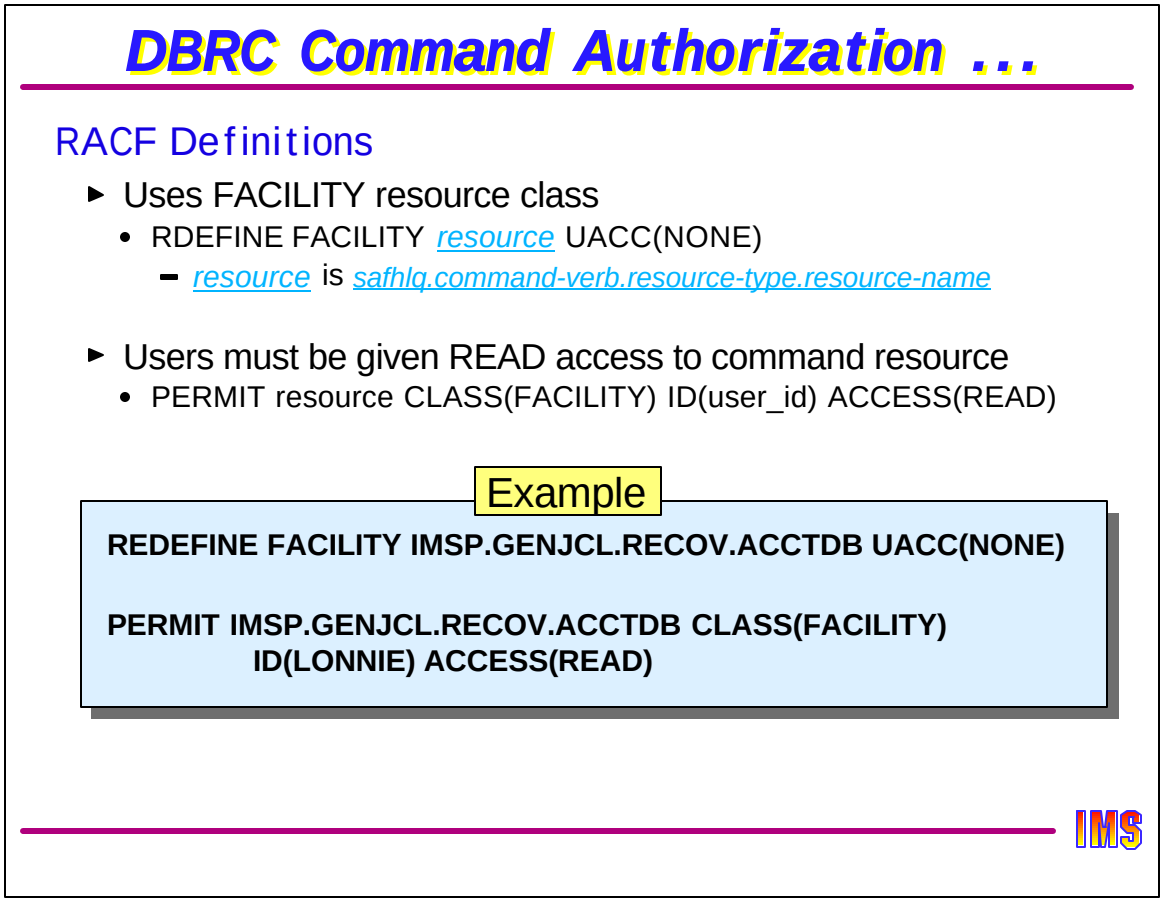
IMS

DBRC command authorization is invoked when the RECONs are set for it. This is done with either the INIT.RECON or CHANGE.RECON command with the CMDAUTH parameter.

The CMDAUTH parameter has two subparameters.

The first subparameter indicates the type(s) of security to be used. SAF specifies that a security product, such as RACF, is to be used for command authorization. EXIT specifies that the DSPDCAX0 exit routine is to be called for command authorization. BOTH specifies that both the security product and the exit routine are to be used. When BOTH is specified, the exit is called after RACF and can override RACF's return. NONE specifies that command authorization is not invoked.

The second subparameter specifies the high level qualifier of the resource name used with command authorization. It must be specified if SAF, EXIT, or BOTH is specified. It may be one to eight alphanumeric characters.



If RACF is used for command authorization, resources must be defined and users must be given authority, or access, to these resources.

The FACILITY resource class is used. The RDEFINE command is used to define resources to be protected. Typically, UACC(NONE) is used. This prevents the use of the command associated with the resource unless a user is specifically permitted to use it.

The PERMIT command allows a user to issue the command associated with a resource. ACCESS(READ) is used for all command authorization processing.

In the example, IMSP is the high level qualifier associated with the RECONs. It was specified as the second subparameter in the INIT.RECON CMDAUTH or CHANGE.RECON CMDAUTH command. The RDEFINE defines the resource associated with GENJCL.RECOV DBD(ACCTDB) or GENJCL.RECOV GROUP(ACCTDB) commands used with these RECONs. The PERMIT specifies that user JONES may issue these commands.

Note that if ACCTDB is a group name, it does not protect individual databases that are part of that group. It only protects the command used with that group name.

DBRC Command Authorization ...

DBRC Command Authorization Exit (DSPDCAX0)

- Optional
- May deny authorization

```
DSP1154I   DBRC COMMAND AUTHORIZATION DENIED BY
DSPDCAX0 FOR USER userid
RESOURCE NAME = hlq.verb.type.name  RC = rc
```

- May be used with security product (RACF, etc.)
  - Security product is invoked first
  - SAF return code and RACF return code/reason code are passed to exit routine
  - Exit may override security product

IMS

The DBRC Command Authorization Exit routine is optional. If used, it must be named DSPDCAX0. If the exit routine denies authorization for a command, message DSP1154I is issued. It includes the non-zero return code from the exit routine.

If both the exit routine and a security product are used, the security product is called first. The exit routine is given the results of the security product call. The exit routine makes the final decision about authorization. It may override the decision indicated by the security product.

Automatic RECON Loss Notification

RECON reconfiguration

- When a RECON fails, the first DBRC to recognize failure copies good RECON to SPARE
- Other DBRCs (e.g. batch or utility running with DBRC) switch the next time they access RECONs
  - *Cannot delete and redefine new SPARE until all DBRCs deallocate bad RECON*
    - May be a long time if in use by batch job
  - No way to force DBRC to access RECONs (except for online DBRC)

Version 8

- When DBRC registers with Structured Call Interface
  - All DBRCs notified as soon as error detected
  - All DBRCs switch to good RECON and deallocate bad RECON
  - Lets user reestablish SPARE quickly
- Structured Call Interface is part of Common Service Layer (discussed later)

IMS

When an I/O error on a RECON or a CHANGE.RECON REPLACE command is issued, DBRC begins a reconfiguration process. This is true for all releases of IMS. The loss of a RECON causes the remaining good RECON to be copied to the spare. This makes the spare a member of the good RECON pair. It also leaves the RECONs without a spare.

IMS V7 added the DSP0388I message. It is sent by the system which begins the RECON reconfiguration process. The message identifies the subsystems using the RECONs. It is intended to be used by operators or automation processes. The message is of the form:
• DSP0388I nnnn SSYS RECORD(S) IN THE RECON AT RECONFIGURATION
• DSP0388I SSID=ssidname FOUND

The installation needs to create a new spare so that it may handle any failure of a member of the new pair. The bad RECON data set name is used for the new spare. The bad RECON must be deallocated from all subsystems which were using it before it may be deleted and redefined. In previous releases each subsystem deallocates the bad RECON the next time it attempts to access the RECONs. At that time, it will discover the change in the RECONs and reconfigure. For batch or utility regions with DBRC, there is no way to "force" the RECONs to be accessed.

In IMS V8, Automatic RECON Loss Notification occurs when all DBRCs have registered with the Structured Call Interface. When this occurs, each DBRC is informed immediately and begins its reconfiguration process, allowing the failed RECON to be deallocated sooner and a new spare to be created. This is a new feature of IMS V8 and will be discussed later when we discuss the Common Service Layer.

DEDB Enhancements

DEDB size

- Support for more than 240 Areas
  - DEDBs can now be defined with up to 2048 Areas
    - 2048 * 4GB = 8TB
  - Just add additional SEGM statements to DBD
- No change to DEDB application interface
- Some migration considerations
  - DEDB randomizer "may" have to be updated to handle larger number of areas passed to randomizer
    - DBFHDC40/44 continue to work without change
  - Log record DSECTs have changed to support 2-byte area number
    - May require changes to user (or vendor) written code
- Fallback
  - Be careful. Cannot fall back to earlier version with more than 240 areas

IMS

Prior to IMS V8, DEDBs could b "partitioned" into a maximum of 240 "AREA"s. Each area is a VSAM ESDS with up to 4GB capacity, or a total database size of 960GB. Even so, some uers found this to be not large enough. Insurance companies, for example, tend to have very large volumes of claims history data. In version 8, DEDBs now support up to 2048 areas, for a total of 8 TB (terabytes). There are no changes to the DEDB externals, but of course it is possible (likely) that 2-stage DEDB randomizers will need to be changed when extending the DEDB beyond 240 areas. Log record DSECTs have also changed to support a 2-byte area number. User programs which use these DSECTs may also need to be changed.

DEDBs with more than 240 areas are not compatible with previous releases of IMS. The user must be careful when extending a DEDB. Once the number of areas is greater than 240, fallback is not possible without reducing the DEDB back to 240 or less areas.

DEDB Enhancements ...

Non-recoverable DEDBs

INIT.DB or CHANGE.DB DBD(dbdname) NONRECOV ...

- Applies to entire database
  - Not area-specific
- Database changes (x'5950') not logged
  - Reduces log volumes
- If area updated during sync interval
  - IMS writes x'5951' log record
  - Used by emergency restart to warn against possible corruption
  - DFS3711W Norecoverable DEDB intergrity warning DEDB=dbdname AREA=areaname
- Especially useful for work databases, scratch pad databases, temporary databases
- Supported for VSO and Non-VSO, shared and non-shared
- Not supported for DEDBs with SDEPs
  - DFS3711A Nonrecoverable DEDB authorization error DEDB=dbdname AREA=areaname

IMS

Non-recoverable full function databases have been supported since IMS V3. Beginning with V8, DEDBs can be defined as non-recoverable. When a DEDB is defined in DBRC as non-recoverable, no x'5950' log records are created when the database is updated. Instead, if a DEDB area is updated during a sync interval, a single x'5951' record is written for each updated area. A x'5951' without a corresponding x'5612' indicates to emergency restart that output thread processing may not have completed and the area may be corrupt. A DFS3711W message is written with the warning, but the area is not stopped. If the AREA is not restored, a U1026 processing error may occur later if a pointer error is detected.

DEDBs with sequential dependents cannot be defined as non-recoverable. If defined as non-recoverable, authorization will fail for each AREA as it is accessed. When this happens, a DFS3711A message is issued and the user must redefine the database as non-recoverable. This will turn on the IC NEEDED flag and an Image Copy will be required.

DEDB Enhancements ...

Shared VSO

- Shared VSO requires coupling facility cache structures
  - User-managed duplexing supported
- V8 now takes advantage of new XES functionality to support
  - System-managed rebuild
    - Move a structure from one CF to another
    - Requires CF Level 9
  - System-managed duplexing
    - Can be used to replace user-manged duplexing now supported for shared VSO areas
    - Requires z/OS 1.2 or later
    - Requires CF Level 11 if CF is 9672
    - Requires CF Level 12 if CF is zSeries
  - Autoalter
    - System may automatically alter structure size; be careful of this one when using non-preloaded areas
  - More discussion on these later

IMS

Shared VSO areas require a coupling facility cache structure to contain the data. Because this is a "store-in" cache structure, committed updates may exist on the structure which do not exist yet on DASD. If the structure fails, then an area recovery would be required. To protect against this, fast path supports user-managed duplexing. This is implemented by defining 2 structures in the CFRM policy and in DBRC. Every update is then written to both structures. If one fails, the other is (hopefully) still available.

User-manged rebuild is not supported for shared VSO structures. This means that to move a VSP structure from one CF to another requires the area to be unloaded (/VUNLOAD AREA command), the CFRM policy changed with a new CF location, and then the area to the restarted.

V8 takes advantage of new functionality in XCF/XES to support both system-managed rebuild and system-managed duplexing. SM-rebuild requires only an update to the CFRM Couple Data Set. Duplexing requires an update to the CDS and to the CFRM Policy. If using SM-duplexing, then user-managed duplexing can be eliminated. These will be discussed in more detail in the Parallel Sysplex Enhancements part of this presentation.

Image Copy 2

IC2 introduced in IMS V6

- Dynamically invokes DFSMSdss to perform "concurrent copy"
  - CC must be supported by control unit (e.g. 3990 or Shark)
- *Minimizes down-time* for databases to perform "clean" image copy
  - /DBR DATABASE
    - Not required if fuzzy image copy wanted
  - Perform logical copy (a matter of seconds)
    - Creates "bit map" of data set tracks in control unit
    - Returns "logical copy complete" message
  - /START DATABASE
    - When track updated, turn on bit, *copy before image to cache*
  - Perform physical copy
    - Copy tracks to output data set
    - If track bit turned on, copy *before image*

IMS

Image Copy 2 was a new utility introduced in IMS V6 to take advantage of the 3990 and Shark "concurrent copy" function to produce fuzzy or clean image copies of a DBDS with a minimum of downtime. IC2 dynamically invokes the DFSMSdss utility to DUMP a DBDS to a tape (or another DASD) data set. To produce a clean image copy, the user had to first DBR the database. The utility would then perform a logical copy. When the logical copy was complete, IC2 would issue a message (DFS3121I) and the MTO could then restart the dataabase for update. The database was offline for only the few seconds that it takes to perform the logical copy.

Several IC2 control card options are used to identify the DBDS being copied, and to determine the options to use when invoking DFSMSdss. But a couple things were missing.
1. Only one data set at a time could be copied
2. The user had no control over the OPTIMIZE option

When only one data set is copied per execution of IC2, the user had to execute several (many) IC2 jobs to copy all of the logically or physically related data sets. Each one would produce the DFS3121I message. It was a user responsibility to make sure that all logical copies were complete before starting any database.

Image Copy 2 ...

Enhanced in V8

- Can copy *multiple database data sets in one execution of IC2*
  - Logical and physical copies performed in parallel
- Can copy *groups of DBDSs* (e.g. DBDSGRP)
  - All DBDSs in group are copied in parallel
  - Single message reports when all logical copies are complete
    - OK to /START databases
- Supports additional *DFSMSdss options*
  - OPTIMIZE 1 |*2* |*3* | 4 (selectable by user)
  - SAMEDS (multiple image copies to same output data set)
- DBRC support
  - GENJCL.IC GROUP(FNCLGRP) SMSNOCIC(C,2) ONEJOB
- ORS support

IMS

There are several enhancements to this process in Version 8.

1. Multiple data sets can be copied in a single execution of IC2. This simplifies the operations and control of this process by allowing multiple related data sets to be copied in a single job.
2. DBDSs can be defined as GROUPs to IC2. When logical copy for all DBDSs in the group are complete, a single DFS3121A message is issued for the entire group, letting operations know that is now OK to restart those databases.
3. Users have more control over the DFSMSdss OPTIMIZE parameter. Prior to V8, this was always set according to the type of IC. OPTIMIZE(1) - copy one track at a time - was set for fuzzy IC and OPTIMIZE(4) - copy one cylinder at a time - was set for clean IC. With V8, OPTIMZE 1,2,3, or 4 can be set in an control card. "2" is copy two tracks at a time, "3" is to copy five tracks at a time.
4. For small DBDSs, it might be desirable to copy multiple data sets into a single output data set. This is referred to as the SAME DATA SET (or SAMEDS) option. This is now supported using the IC2 control card.
5. GENJCL.IC has been enhanced to create the JCL and control cards for a GROUP of DBDSs defined to DBRC as a DBGROUP or DBDSGROUP.

Additionally, ORS has been enhanced to supported all of the IC2 produced image copies, including SAMEDS.

Image Copy 2 ...

Example Logical Copy complete notification

- /DBR DATAGROUP FNCLGRP
- GENJCL.IC GROUP(FNCLGRP) COPIES(2) SMSNOCIC(C,2) ONEJOB

```
DFS3121A LOGICAL COPY COMPLETE FOR GROUP FNCLGRP; 0 OF 5 DATA SETS FAILED
DFS3121I COPIED DB/AREA EMPL DDN ACCT1 DSN IMSPRD.DB.ACCT.ACCT1
DFS3121I COPIED DB/AREA EMPL DDN ACCT2 DSN IMSPRD.DB.ACCT.ACCT2
DFS3121I COPIED DB/AREA CUST DDN CUSTA DSN IMSPRD.DB.CUST.CUSTA
DFS3121I COPIED DB/AREA CUST DDN CUSTB DSN IMSPRD.DB.CUST.CUSTB
DFS3121I COPIED DB/AREA CUST DDN CUSTC DSN IMSPRD.DB.CUST.CUSTC
```

- /START DATAGROUP FNCLGRP
  - Resume updating databases

IMS

This shows examples of logical copy completion messages sent to SYSPRINT. Only the DFS3121A messages are sent to the system console.

In this example, two databases with a total of five data sets are to be copied as a GROUP defined to DBRC as FNCLGRP. The group is first DBRed. GENJCL.IC is then issued to generate the appropriate IC2 JCL for the group. In this example, we are creating 2 clean image copies and invoking the COMPRESS option (added in IMS V7) and requesting OPTIMIZE(2).

Logical copies of all 5 data sets are done in parallel. When they are ALL complete, the DFS3121A message is issued indicating that it is OK to start those databases. There is also a DFS3121I message for each DBDS for which logical copy was successful. This may be especially interesting if logical copy failed for one or more DBDSs.

Image Copy 2 ...

Stacked versus SAMEDS

- Stacked
  - Multiple output data sets on single volume
  - Logical copy for 2nd DBDS not done until physical copy for 1st DBDS complete
    - Even if output volume is DASD
    - DFSMS restriction
  - Logical and physical copy serial
    - Logical copy complete message delayed
- SAMEDS
  - Single output data set
  - Logical copy for all DBDSs in parallel
  - Physical copy for all DBDSs serial

IMS

There is a difference between STACKED and SAMEDS image copies. STACKED means that multiple IC data sets are to be written to a single output VOLUME. But there are still multiple data sets. The problem with this is that the logical copy for the second data set is not done until the physical copy for the first is complete, compromizing the availability advantages of IC2.

SAMEDS means that there is only a single output data set with all of the DBDSs in that data set. Logical copies for all DBDSs is done in parallel at the beginning of the process. Header records identify which DBDSs the following IC records are for.

In both cases, physical copy is serial.

Online Recovery Service (ORS)

ORS is an IBM IMS Tool (5655-E50)

- Recovers IMS databases quickly and easily
- Fast processing
  - *Parallel reads* of inputs
    - Image copies, change accumulations, and logs
  - Logs and Change Accums are *read once* for all database data sets
    - Saved in data space until needed
  - *Parallel writes* of outputs
    - Databases recovered in parallel
- Change accumulation not required for data sharing
- Point-in-time recovery to *any time*

IMS

ORS is an IBM product used in conjunction with IMS. It requires IMS V7 or later. ORS recovers databases quickly by using parallel processing of inputs an outputs. It reads inputs only once, even if they are needed for the recovery of multiple data sets. ORS can merge data sharing logs. This eliminates the need to run Change Accumulation before recovering databases. ORS can do time stamp recoveries to any time. This is called Point in Time Recovery (PITR). PITR eliminates the need to /DBR databases to create recovery points. A PITR recovers all updates that were committed before the specified time.

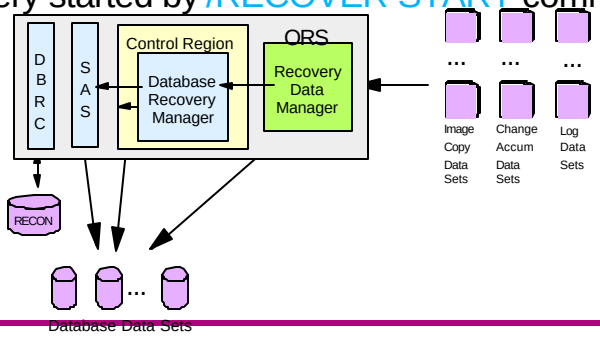
Online Recovery Service ...

A control region function

- Invoked by IMS commands
- Executes in parallel with other online activity
- May run on any control region where the DBs are defined

Recovers a list of database data sets

- List is built by /RECOVER ADD xxx commands
  - Command may add a data set, a database, or a group
- Recovery started by /RECOVER START command



IMS

ORS is an online system function. It is invoked by the /RECOVER command. ORS may run on any control region where the databases are defined. Since ORS uses DBRC, the control region must have access the the RECONs where the databases are registered.

ORS recoveries are invoked in two steps. First, the database data sets to be recovered are added to a list. This is done with the /RECOVER ADD command. Since DBRC groups may be specified, often only one /RECOVER ADD command is required. Once the list is built, the recovery is initiated with the /RECOVERY START command.

ORS Enhanced Capabilities

Most ORS enhancements do not require IMS V8

- Applicable to IMS V7 and IMS V8

Enhanced functionality supports ...

- Stacked image copies
  - Suppresses rewind between files
- IC2 SAMEDS image copies
  - Single restore request for all image copies
- "Image Copy Extensions" compressed image copies
  - Decompresses image copy records
- Virtual tape caching
  - Takes advantage of Tape Managment System caching
  - Can explicitly exploit two TMSs concurrently (caching primary/secondary or RLDS/SLDS logs)
- Messages to IMS MTO (requires IMS V8)
  - ORS message embedded in DFS4299I message

IMS

ORS has been enhanced in IMS V8 to support the following:

- Stacked image copies without rewinding the tape between files. ORS restores the ICs in file sequence number order.
- ORS supports IC2 SAMEDS image copies by issuing a single RESTORE request for all ICs in the data set.
- Image copies compressed using the IMS Image Copy Compression Tool will be expanded by ORS by invoking the tool for expansion during IC restore.
- ORS has added Tape Management System (TMS) support. A TMS can move files from tape to DASD in anticipation of the use of these files. This can shorten times required to process these files. It also allows multiple files that are stored on one tape to be processed concurrently. ORS has added TMS caching support for log data sets. The logs are cached to DASD while ORS is restoring Image Copies. The application, ORS in this case, issues tape reads. The TMS accepts these reads and satisfies them from its DASD cache.
- ORS messages (FRDnnnn) are now sent to the MTO. The ORS message itself is imbedded in the DFS4299I message.

Support for Stacked Image Copies, Compressed Image Copies, and Virtual Tape Caching are also available in IMS V7 by APAR.

IMS/DB2 Coordinated DR

Synchronization of IMS and DB2 logs at remote site

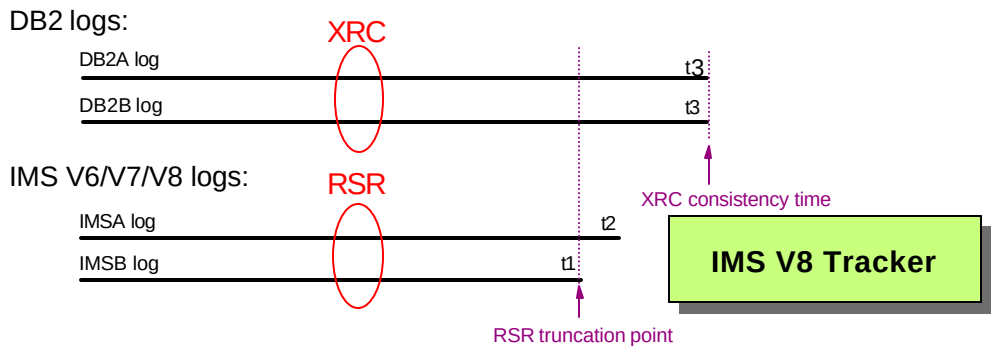
- Environment
  - IMS with RSR
    - IMS logs are sent to remote site using RSR
    - Optionally, IMS databases may be "shadowed" at remote site
  - DB2 *data sharing systems* using XRC (eXtended Remote Copy)
    - DB2 logs and boot strap data sets (BSDSs) sent to remote site using XRC
    - *Databases not included*
- Allows IMS and DB2 databases to be recovered to same time
  - Especially applicable to environments with limited bandwidth
    - DB2 updated CIs not sent to remote site - just the logged changes

IMS

IMS V8 provides support to coordinate IMS and DB2 data when a disaster recovery is done. This support applies to environments where RSR is being used for IMS and XRC is being used for DB2 logs and BSDSs. DB2 data sharing must be used because the support relies on timestamps in the DB2 log. Without data sharing, DB2 would use RBAs, not timestamps. The support makes is easy to recover the IMS and DB2 databases to the same time.

This support is especially applicable to environments with limited bandwidth for XRC transmissions. These environments cannot support the tranmission of updates for all of the DB2 databases. Instead, only DB2 and IMS logs and DB2 BSDSs are transmitted.

IMS/DB2 Log Synchronization



- IMS RSR tracking subsystem truncates IMSA's log at time **t1**
  - The truncation point timestamp is displayed in DFS2933I message
- IMS log truncation timestamp is supplied as the "to timestamp" for conditional restart of DB2A and DB2B
  - DB2 restart truncates the logs for DB2A and DB2B at time **t1**
- Result : DB2 and IMS logs end at same time; allowing consistent timestamp recoveries

IMS

Disaster recovery coordination between IMS and DB2 is done by synchronizing their logs.

This shows an example of synchronizing DB2 and IMS logs using the new capabilities. In this example, both IMS and DB2 are doing data sharing. There are two DB2 subsystems and two IMS subsystems. Each subsystem is producing its own log stream.

The DB2 logs are synchronized by XRC but they end at time t3. The IMS logs are kept behind the DB2 logs. The IMS logs may end at different times. When a takeover occurs, IMS RSR at the tracking site truncates the IMSA log from time t2 to time t1. It writes a message with time t1. This time is used as the "to timestamp" for DB2 conditional restart of the two DB2 systems. This is the LRSN value. This causes the DB2 logs to be truncated at time t1. So, all logs have the same end point, time t1.

This allows timestamp recoveries of the IMS and DB2 databases to be done to the same time. Actually, IMS timestamp recoveries are not required if database shadowing is done for them.

Systems and Transaction Manager

TM Enhancements

- APPC
- Transaction Trace
- MSC

Systems Enhancements

- Syntax Checker
- Parallel DB Open
- Dynamic SLDS allocation
- VSCR

IMS

The next parts of this presentation address changes related to the transaction manager (i.e. not DBCTL) or to the IMS system itself (these do apply to DBCTL).

Dynamic APPC Descriptors

Ability to Add/Delete LU 6.2 descriptors without an IMS restart

- To **ADD** a new LU 6.2 Descriptor:
  - Create new descriptor member in IMS Proclib
    - DFS62DTy where y is a user-defined suffix
    - Include new descriptors in new member
  - Issue **/START L62DESC *y***
    - Where ***y*** is the suffix associated with the new member in Proclib
    - Proclib members are "additive"
- To **DELETE** an LU 6.2 Descriptor:
  - Issue **/DELETE L62DESC *descriptor***
    - Where ***descriptor*** is the name of the one to be deleted
- Changes are not saved across a restart
  - Update ***DFS62DTx*** member used during IMS initialization

IMS

LU 6.2 descriptors allow system programmers to associate an application-specified destination name, e.g., ALTPCB destination name, with an LU 6.2 application program. For several releases of IMS, support has been provided to allow information in a descriptor to be dynamically modified via a /CHANGE DESCRIPTOR command but this assumed that the descriptor was created during IMS initialization. Adding or deleting descriptors required a restart of IMS.

IMS V8 adds support to dynamically create or delete descriptors during the execution of an online system.

To add a descriptor, first create a new member in proclib called DFS62DTx where x is a suffix other than the IMS nucleus suffix (SUFFIX= on IMSGEN macro). DFS62DTs (where "s" is the nucleus suffix) is the member used during IMS initialization and contains all the LU 6.2 descriptors that are already part of the system. Once this is done, the command "/STA L62DESC x" reads the new member in PROCLIB and builds the descriptors that are defined in the member. The same messages that are displayed during IMS initialization are also used during the dynamic addition of descriptors.

To delete a descriptor, the "/DEL L62DESC descriptor" command specified the names of the descriptors to be deleted.

Note that the effect of the /STA and /DEL commands are applicable only during the life of an IMS system and are not carried across a restart. To ensure that the appropriate descriptors are added or deleted for the next restart, update the DFS62DTx member used during IMS initialization, i.e., where x is equal to the IMS nucleus.

CPU Time Limit for CPIC Transactions

CPU Time Limit for CPI-C driven transaction

- CPUTIME = 0 - 1440
  - Specifies the number of CPU seconds before a timeout occurs
    - 0 = no timeout (default)
    - 1440 = maximum value
  - If time is exceeded
    - Abend U0240
    - Message DFS554x
  - TP_Profile can be dynamically updated

- Protects against program loops
  - Does not protect against calls waiting for an APPC response

IMS

If an application program loops, IMS resources are held until the transaction can be terminated. During this time any resources held by the application, e.g., database locks, the dependent region, etc., are unavailable to any other application. Many installations have addressed this situation for non-CPIC transactions by specifying a timeout value in the PROCLIM parameter of the TRANSACT macro. Since CPIC transactions are not defined to IMS, this specification is not applicable. IMS V8 addresses this area for CPIC transactions with a new TP_Profile specification for CPUTIME.

The following is an example of an APPC/MVS TP_Profile entry for an IMS CPIC transaction:

```
TPADD TPSCHED_EXIT(DFSTPPE0)
TPNAME(CPICLOOP4)
SYSTEM ACTIVE(YES)
TPSCHED_DELIMITER(##)
TRANCODE=CPICTRAN
     CLASS=1
     MAXRGN=1
     CPUTIME=25
##
```

When the time specified in CPUTIME is exceeded when processing a message, the CPIC transaction is abended in a way consistent with non-CPIC transaction timeouts. The transaction gets a U0240 abend which is accompanied by a DFS554x message.

APPC/MVS continues to allow dynamic updates and activation of TP_Profile entries.

Note that this capability protects against situations such as program loops that incur CPU usage. Other waits such as waiting for an APPC call (receive_and_wait) to complete are not affected.

Outbound LU Support

Outbound conversations require active APPC/IMS LU

- Default is the LU defined to APPC/MVS as BASE
  - If "disabled", outbound conversations cannot be established
    - Even if there are multiple APPC/IMS LUs

IMS Outbound LU support enhancement

- Ability to specify which = LU is to be used for outbound conversations
  - Assumes IMS has multiple APPC/IMS LUs defined in APPCPMxxx
    - Default continues to be the BASE
  - **DFSDCxxx** parameter **OUTBND**=*luname*
  - **/CHANGE APPC OUTBND** *luname*
- Restart reverts to OUTBND value in DFSDCxxx or BASE LU

/DISPLAY APPC

- Shows: TYPE=OUTB | BASE

IMS

Prior to IMS V8, APPC/IMS outbound conversations always used the APPC/IMS LU defined to APPC/MVS as the BASE LU. If this LU was disabled or unavailable, outbound conversations could not be allocated even if there were other APPC/IMS LUs defined for this IMS system that were active and available.

IMS V8 allows specification of an APPC/IMS LU, other than the BASE, to be used when establishing new outbound conversations. This presumes that the APPCPMxxx member of the MVS PARMLIB has multiple APPC/IMS LUs defined and associated with this specific IMS system.

The LU defined as BASE is the default and continues to be used unless an override name is specified in the OUTBND parameter in DFSDCxxx. At any time during the execution of an online system, the /CHANGE APPC OUTBND command can be issued to specify a different override. During a restart, IMS always reverts to the specification in DFSDCxxx unless it is not defined, in which case, the default goes back to the BASE LU.

The /DISPLAY APPC command can be used to list all the APPC/IMS LUs defined to the system and which one is being used for outbound processing and/or the BASE.

Transaction Trace (TT) - Basics

A *facility* provided by OS/390 and z/OS

Shows the flow of work between subsystems that combine to service a transaction

- Provides ability to ***track a work unit*** across multiple subsystems in the ***same sysplex***
- Facilitates debugging problems in multi-system application environments
- Ties work unit together using transaction trace token

IMS V8 may exploit

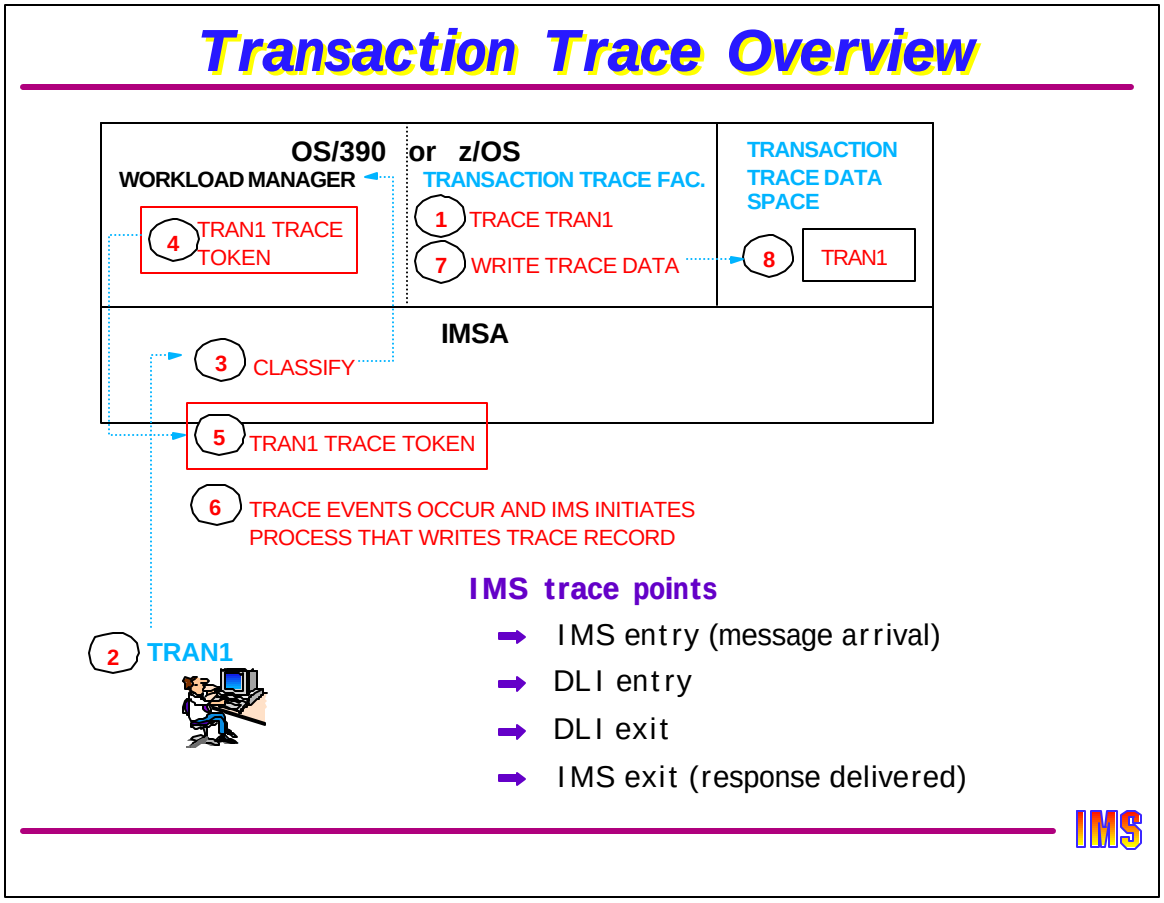
- Transaction trace facility of OS/390 or z/OS
  - Continues to have the standard IMS transaction trace support
- Is the first (and so far, the only) subsystem that supports TT

IMS

Transaction trace is a facility of OS/390 and z/OS operating systems. It provides a consolidated trace of key events for the execution path of application or transaction type work units running in a multi-system application environment.

By tracing the path of a work unit running in a single system, or (more importantly) across systems in a sysplex environment, that is being processed by multi-system transaction servers, subsystem interfaces, and resource managers such as IMS Version 8, transaction trace helps a system programmer debug problems in those environments.

As of now, transaction trace is supported only by the IMS V8 subsystem. For work that involves multiple subsystems, such as an IMS transaction executing DB2 calls, only the IMS calls will be traced.



The diagram provides an illustration of some basic concepts on how transaction trace works. Once the transaction trace facility is activated, IMS transactions (such as TRAN1 in the diagram) may be traced.

Workload Manager CLASSIFY processing determines whether the current work unit (transaction) should be traced. If tracing is required, a non-zero token is built and returned to IMS (the CLASSIFY) caller. The transaction trace token is set to zero if no tracing is to be performed for that work unit. IMS propagates the token in a manner similar to the propagation of the service class token.

IMS Version 8 has been enhanced to issue new macros to:

- Determine if tracing can be performed
- Initiate the writing of a transaction trace record

Transaction trace writes data in a transaction trace data space. The MVS DUMP command is used to dump the transaction trace data space. If an external writer has been defined, transaction trace records are also written to the external writer. Interactive Problem Control System (IPCS) product may be used to view the transaction trace records.

MSC FICON CTC Support

Enhances MSC to take advantage of FICON CTC support

- Greater bandwidth
  - Increases volume of messages between IMS systems
  - Increases the distance between IMS systems
- Activated by changing the IMS online procedure
  - DD cards specifying the CTC address for the link should be changed to specify the CTC FICON address
    - IMS detects a FICON CTC and invokes the support
  - Note: No required IMS system definition changes or sysgen
    - Assumes the existing link is already defined as CTC

IMS

The IMS MSC FICON CTC support increases the volume of IMS messages that can be sent between IMS systems when using the IMS MSC facility. This capability takes advantage of the CTC bandwidth enhancement provided by the FICON channel support for CTCs on the new IBM zSeries processors. It is estimated that one FICON CHPID can do the work of a number of ESCON CHPIDs. This is the result of the fast data transfer and higher throughputs of FICON. The distance between hosts can also be increased.

IMS MSC FICON CTC support requires at least one side of the MSC link be an IBM zSeries with the FICON channel and FICON CTC microcode. The other side can be any processor with a FICON channel.

An IMS sysgen is not required to convert existing CTC links to FICON. This is because the TYPE parameter on the MSPLINK macro continues to be specified as CTC. The support is activated by changing the IMS online procedure DD cards from specifying the CTC address to specifying the FICON addresses (ie: //DDNAME DD UNIT=xxxx where xxxx = the CTC FICON address). IMS detects that the address is a FICON CTC and invokes the MSC/FICON support.

This capability was added to IMS V7 as APAR PQ51769.

Syntax Checker

Syntax Checker is a new IMS ISPF application

Its primary functions are to

- Define, verify, and validate PROCLIB member DFSPBxxx parameters and value specifications prior to (re)starting IMS
  - For example, Syntax Checker ...
    - Reads and displays a Proclib member's parameters and values
    - *Verifies that parameters and values are valid*
    - Allows user to add/delete parameters and/or modify values
    - Allows user to save to same or different Proclib member
- Identify new and obsolete parameters
  - Useful when migrating to new version of IMS
- Provide detailed online help text at the parameter level

IMS

The IMS Syntax Checker is a new ISPF application delivered with IMS V8. The Syntax Checker only runs in an ISPF environment on TSO/E.

Syntax Checker provides the capability to define, verify, and validate the parameters (and their value specifications) of the DFSPBxxx member of IMS.PROCLIB before shutting down and restarting the IMS control region to activate the startup parameters. The intention is to expand this support to the other IMS proclib members.

Syntax Checker also provides detailed online help text a the parameter level.

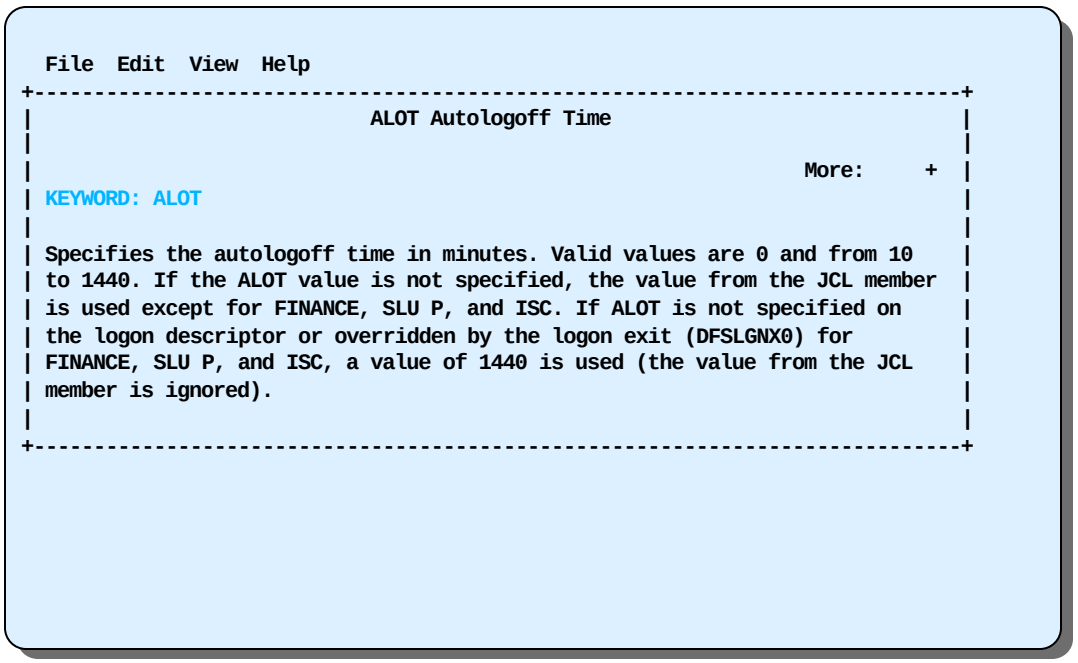
Invalid Parameter

```
  File  Edit  View  Help
-------------------------------------------------------------------------
                  IMS 7.1 Parameters for DB/DC
Command ===>
DFSI920 Parameter value invalid
Press enter to check the syntax.

Data Set Name  . . : IMS71.IMS1.PROCLIB(DFSPBIMS)
IMS Release  . . . : 7.1

  Sel Codes: C = Comment D = Delete / = Select

 Sel Keyword    Value            Description
  _  ________ = ________
  _  ALOT    = 9                ETO Auto Logon Off Time
  _  AOIS    = A                ICMD Security Option
  _  APPC    = Y                Activate APPC/IMS (Y|N)
  _  APPLID1 = IMS1             VTAM Applid of Active IMS System
  _  APPLID2 = IMS2             VTAM Applid of XRF Alternate System
  _  CHTS    = 1000             Number of CCB Hash Table Slots
  _  CMDMCS  = N                MCS/EMCS Command Option: N|Y|R|C|B
  _  DBBF    = 1000             Number of Database Buffers
```

IMS

In this example, the ALOT parameter has an invalid value. The help screen for ALOT can be invoked by placing the cursor on the ALOT line and pressing PF1.

Keyword ALOT Help



IMS

To obtain the help screen place the cursor on the line containing the value and hit the PF1 key. Note the help text that is displayed for the ALOT keyword that contained an invalid value on the previous visual.

Not Valid in Release

```
  File  Edit  View  Help
-------------------------------------------------------------------------------
                  IMS 8.1 Parameters for DB/DC
Command ===>
DFSI926 Keyword CHTS not valid in Release 8.1
Press enter to check the syntax.

Data Set Name  . . : IMS71.IMS1.PROCLIB(DFSPBIMS)
IMS Release  . . . : 8.1

  Sel Codes: C = Comment D = Delete / = Select

 Sel Keyword    Value           Description              More: -
  _  CHTS   =   1000            Number of CCB Hash Table Slots
  _  CMDMCS  =  N               MCS/EMCS Command Option: N|Y|R|C|B
  _  DBBF    =  1000            Number of Database Buffers
  _  DBFX    =  10              Num. DB Buffs available at FP Reg Start
  _  DBRCNM  =  DBCPROC         DBRC Problib Member Name
```

IMS

This is an example of a parameter which was valid in IMS V7 but is no longer valid in IMS V8. The CHTS (Conversation Hash Table Slots) parameter is not used by V8 and must be removed before bringing up V8. Note that this was a V7 DFSPBIMS member. The user entered 8.1 in the IMS Release field to determine whether the parameters were valid for IMS V8.

Database Allocation/Open/Close

Full function database authorization, allocation, and open processing prior to V8

- Database authorization and DBDS allocation
  - Occurs during program specification block (PSB) schedule
  - Incurs 1 DBRC authorization call per database
- Database data set open
  - Occurs at first DL/I call requiring the data set

Impact

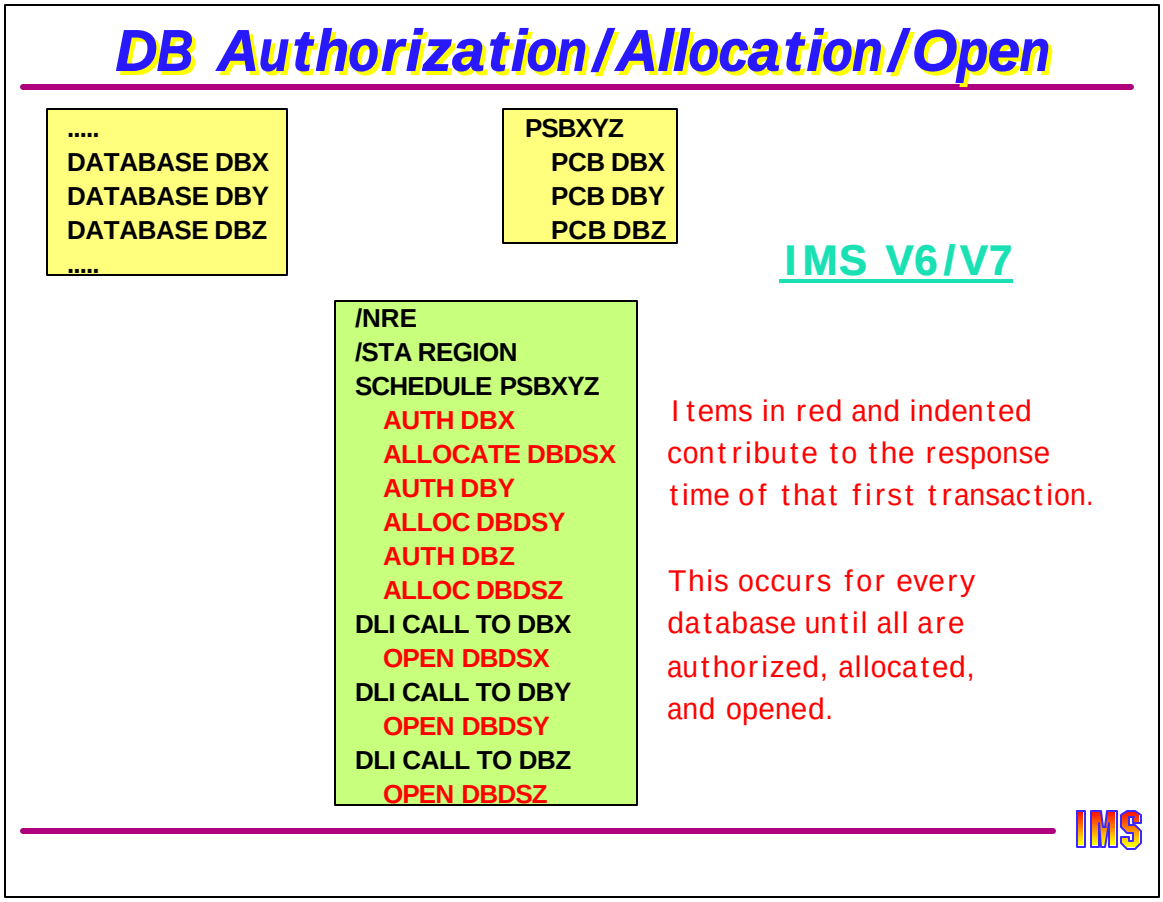
- Large number of concurrent allocation requests
  - Potential increase in RECON contention
  - Potential increase in length of time needed to achieve steady state
- May affect response times for early end-users

IMS

At the completion of normal or emergency restart, databases and data base data sets are not authorized, allocated, or opened. ERE will allocate, open, and close DBDSs as needed for dynamic backout, but will not leave them in that state when ERE completes.

Full function databases are authorized and allocated when a PSB which has a PCB for that database is scheduled. Database data sets are opened the first time a call requires that data set. This is typically during the scheduling or execution of a transaction and has several negative impacts.

1. There are frequently a large number of these activities going on as soon as the IMS online system becomes available to end-users.
2. Each DBRC request was serial, one database at a time. Allocation and open were also serialized on a TCB in the control region.
3. End-users had to wait for this processing to complete to get their first transaction executed.



This chart shows the activities required when PSBXYZ if first scheduled (as the result of an end-user entering TRANXYX). Each database is individually and serially authorized, allocated, and opened.

Parallel Auth/Alloc/Open

For full function IMS databases

- During warm or emergency restart
  - Perform database authorization, allocation, and open processing for *all databases with data sets allocated at time of last shutdown*
    - Not part of PSB schedule or DL/I call processing

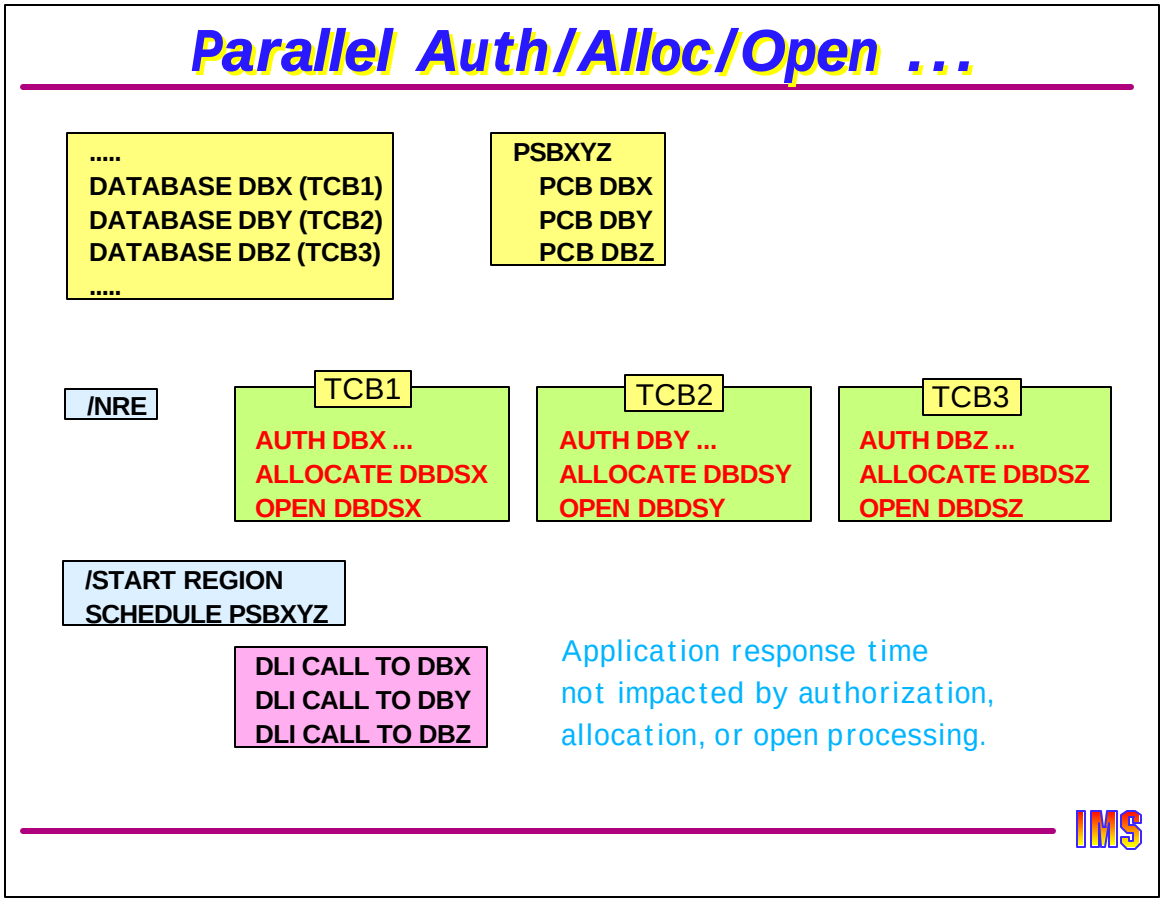
Ten parallel TCBs for database processing during IMS warm or emergency restart, and during IMS shutdown

- Each databases assigned to one of *10 TCBs* for authorization and open/close/eov processing
  - Assignment based on local DMB number
- During warm or emergency restart
  - Each TCB, in parallel, ...
    - Issues single DBRC authorization request for all DBs assigned to that TCB
    - Performs DBDS ALLOCATION and OPEN processing

IMS

In IMS V8, database authorization and data set allocation and open, are done during warm or emergency restart for all databases and their respective data sets that were open at the time of the previous shutdown.

Ten new TCBs have been created to perform these functions in parallel. Databases are assigned algorithmically to one of the 10 TCBs. In general, each TCB will handle about 1/10th of the databases. One DBRC call will be issued by each TCB with a list of all "assigned" databases. When authorization is complete, each TCB will then proceed to allocate and open each DBDS. Note that ALL DBDSs for a database will be allocated and opened, even if not all DBDSs were open at the time of the prior shutdown.

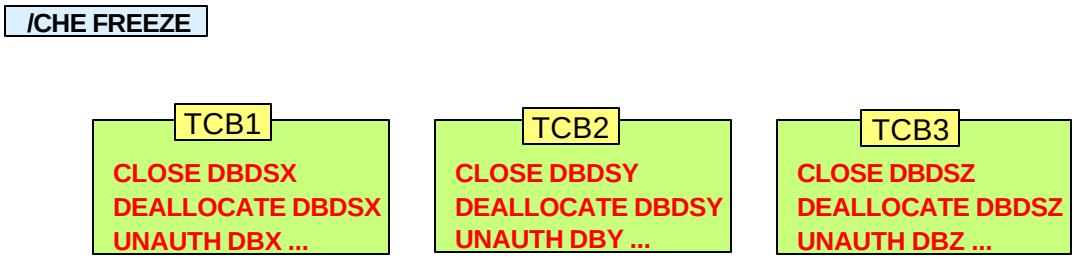


This chart shows how the authorization, allocation, and open process are done in parallel under one of 10 TCBs (only 3 are shown). Overall, even though DBRC authorization processing will process each request from IMS serially, the overall impact is to improve the elapsed time of the entire process and to do it before end-users submit their work.

Parallel Close/Dealloc/Unauth

During shutdown

- New TCBs process shutdown for assigned databases
  - Single DBRC request (per TCB) to unauthorize databases
  - Parallel close and deallocation of database data sets



IMS

The same 10 TCBs are used at IMS termination for close, deallocation, and unauthorization processing, making IMS shutdown quicker.

Online Reading SLDS After Restart

Online may need to read SLDS after restart

- Previous releases could read SLDS only during emergency restart
  - Needed if backout records or shared queues records no longer available from OLDS
- IMS V8 also reads SLDS if needed for dynamic backout or shared queues
  - SLDSs dynamically allocated as needed
    - Even if required after restart completes
  - SLDS log records are stored in data space(s)
- Enhancement also available for IMS V7
  - APAR PQ50657
- New commands
  - /STOP (/START) SLDSREAD disables (enables) function

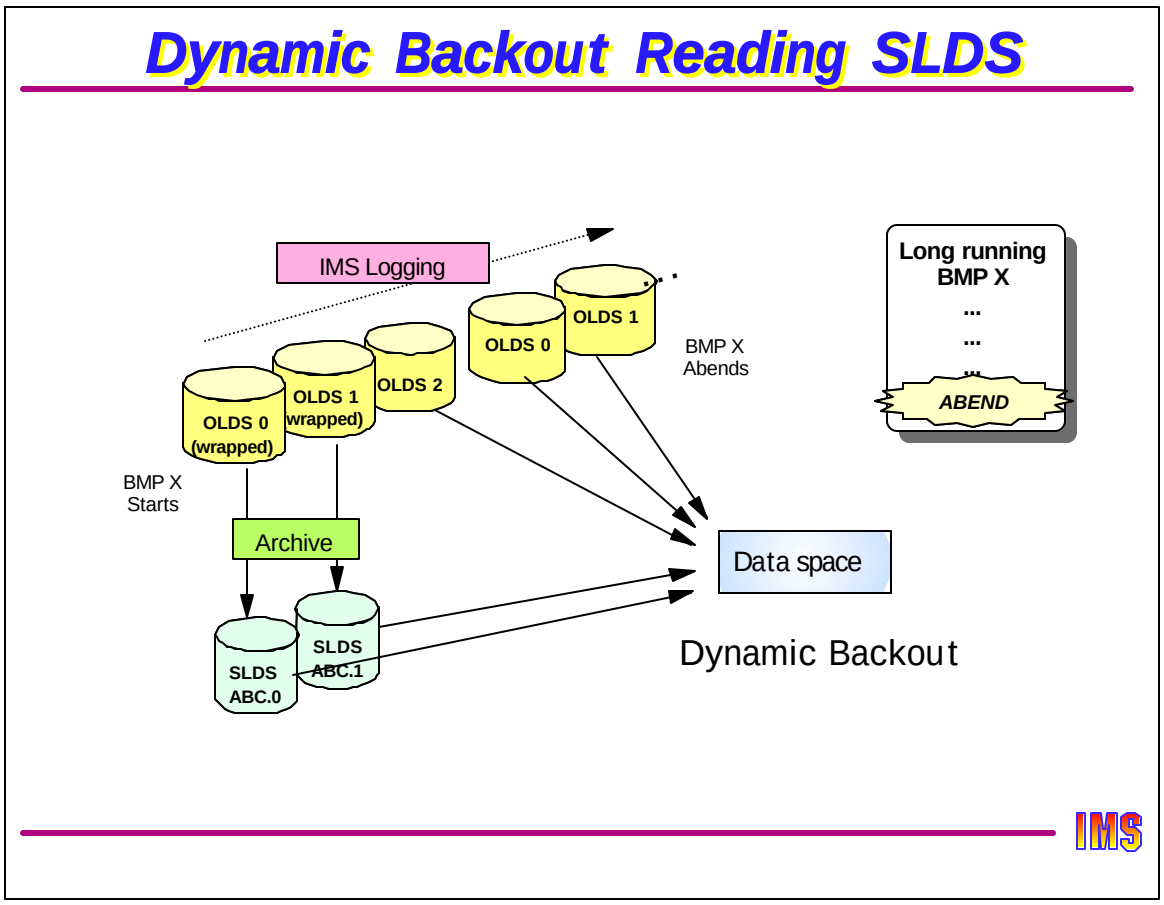
IMS

In previous releases an IMS online system would read an SLDS only during restart processing. This was done when the log records needed to backout work that was in flight when the system failed were no longer available from an OLDS. That means the OLDS had been archived and either reused or deleted. IMS would not read the SLDS after restart processing had completed. If dynamic backout needed a record that was only available on a SLDS, the backout failed and the user was required to run the batch backout utility. This was a rare occurrence, but it could happen. Usually, it was the result of the abend of a BMP which had run a very long time without taking a checkpoint. Records needed for its backout were written to OLDS which had been archived and reused. The enhancement in IMS V8 addresses this problem.

If a log record is needed for a dynamic backout and the log record is no longer available on an OLDS, IMS will dynamically allocate the required SLDS(s). If will allocate the SLDSs is reverse time sequence. That is, the most recent SLDS containing records no longer on a OLDS will be allocated first. IMS reads and saves the log records from this SLDS. The records are saved in a data space. If this SLDS does not contain all of the required records, additional SLDSs are allocated, read, and saved. This continues until all the required records have been read.

This enhancement also applies to shared queues. If the message queue structure and its overflow structure becomes full, committed output messages are logged but not written to the structures. These messages were created before the structures became full, but not committed until after the full condition occurred. If space in the structures is available at a later system checkpoint, these messages are written to the structures. The log records have to be read to do this. If the log records are no longer available from the OLDS, IMS V8 will dynamically allocate the required SLDSs to read these log records.

This enhancement is also being made available in IMS V7 through APAR PQ50657.



This is an illustation of the use of the new capability. In this example, a long running BMP begins when OLDS0 is in use. As the BMP runs, its log records are written to OLDS0, OLDS1, and OLDS2. When a new OLDS is used, the previously used OLDS is archived. When OLDS0 and OLDS1 are reused, the log records for the BMP are written to them. The BMP abends when it is logging to OLDS1. In previous releases, the dynamic backout would fail since the log records originally written to OLDS0 and OLDS1 are no longer available. This failure does not occur with IMS V8. IMS V8 reads the archived logs, that is the SLDSs, for those OLDS that have been reused. In this case, it reads SLDS ABC.0 and SLDS ABC.1 to get the records originally written to OLDS0 and OLDS1. The log records are placed in one or more data spaces.

RRS Support for IMS Batch

In a Stand-alone IMS Batch environment

- IMS V8 now provides full coordinated two phase commit with attached subsystems
  - DB2 V6 or MQ V5.2
- Batch program requires connection to RRS
  - New JCL PROC parameter “RRS=Y”
  - RRS uses System Logger for saving coordination data
- Batch program requires DASD logging and BKO=Y

Utilized by IMS DataPropagator V3R1

- Uses MQ to give “asynchronous near real time” data propagation for Batch DL/1

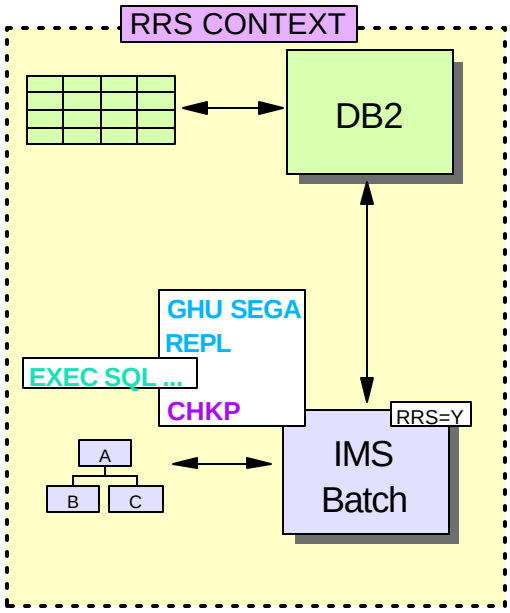
IMS V7 support with APAR PQ51895

IMS

Batch Resource Recovery Service (RRS) support allows batch programs to use MQSeries with coordinated commit. It also provides for a full two phase commit for batch programs accessing DB2 as well as IMS DB, where today's Batch Attach from DB2 does not support coordinated commit. And it also allows for work which captures data and propagates it to another system (DB2) to participate in the 2-phase syncpoint process along with the IMS work, thus making sure that all the work is done or not done where it is all part of the same unit of recovery.

Batch RRS support is also being utilized by the new IMS DataPropagator V3R1 which provides asynchronous, near real-time IMS-to-DB2 Propagation by taking advantage of the asynchronous messaging functions of the IBM MQSeries product. With this the operations and administration of an asynchronous propagation is simplified, easier to use, and less error-prone. This near real-time propagation would also offer improved performance and reliability, minimizing the impact on mission-critical IMS applications that are updating IMS DB data being propagated to DB2.

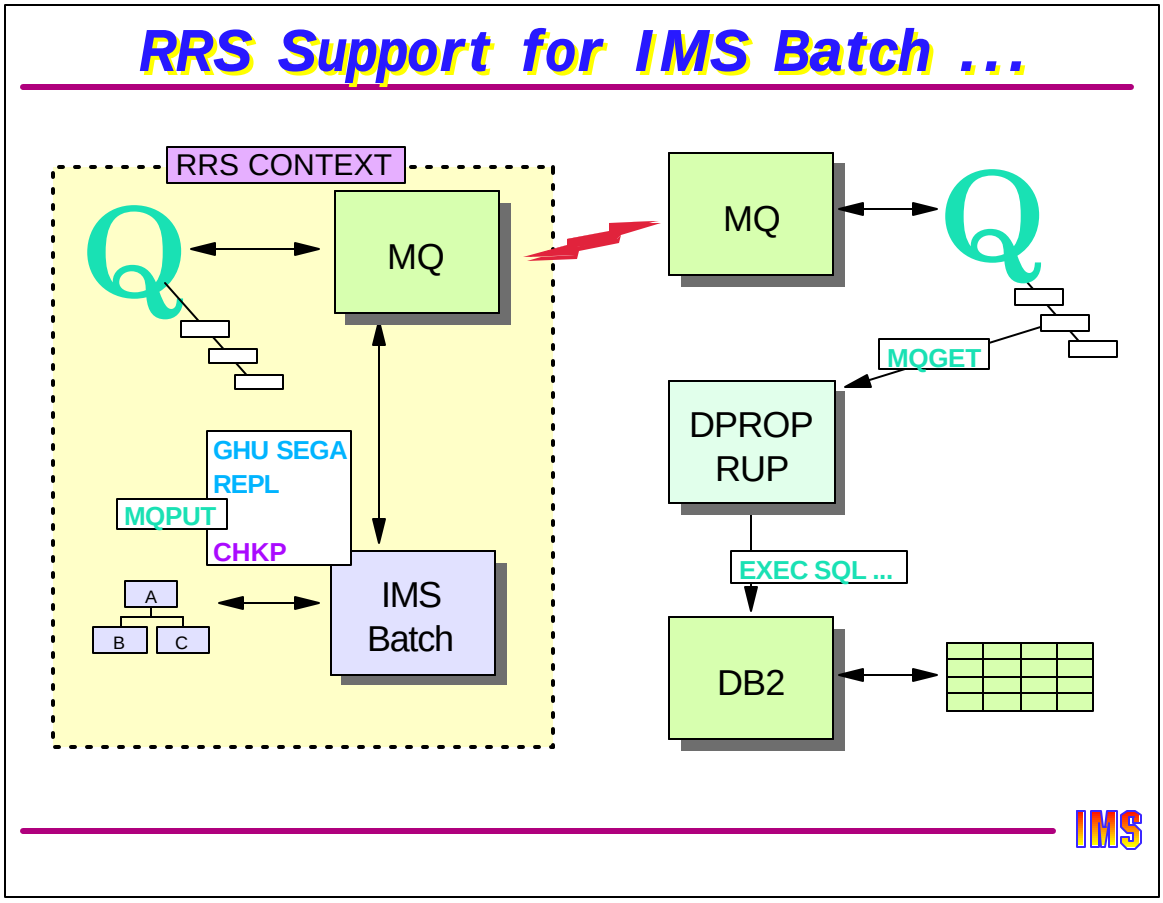
RRS Support for IMS Batch ...



- IMS and DB2 register with RRS
- IMS updates IMS DB
- IMS expresses interest in UR
- Data capture exit updates DB2 (may be DPROP)
- DB2 expresses interest in UR
- CHKP - initiate 2-phase commit for UR
- RRS manages 2-phase commit

IMS

This is an example of an IMS Batch region (not BMP) using the DB2 batch attach function to update both IMS and DB2 databases. RRS must be active for coordinated two-phase commit to occur. Both IMS and DB2 register with RRS. When IMS updates a database, it registers interest in the "Unit of Recovery" which is being managed by RRS as part of the "context" (a sync interval in IMS terms). When a DB2 resource is updated, DB2 registers interest in the same UR. When the IMS program commits (CHKP), RRS is invoked to manage the 2-phase commit process between IMS and DB2.



In this example, DPROP is being used in an IMS Batch application to propagate IMS DB chagnes to DB2 "asynchronously." It does this by capturing the IMS changes (IMS Data Capture Exit) and sending a message with the changes to DB2 using MQSeries. Both IMS Batch and MQSeries must be using RRS. When IMS Batch commits, MQSeries will "commit" the messages, which can then be sent to wherever the Dprop Relational Update Program is running. The RUP then issues an MQGET to retrieve the message and update DB2. The only part of this that is within the 2-phase commit scope is the IMS update and the queuing of the message by MQSeries.

IMS Java

Write, compile, and run IMS programs written in Java

- *Development environment*
  - IMS Java Class Library
    - Set of packages that contain groups of classes
  - Java classes are pre-written reusable code that can be invoked by Java program to ...
    - Access IMS message queues and databases
    - Support APIs familiar to Java programmers
- *Runtime environment* supports new dependent region types
  - JMP and JBP (V7 APAR PQ53944)
    - Persistent reusable Java Virtual Machine

Benefit

- Incorporates Java programming model into IMS environment
- Allows Java programmers to code IMS applications
  - Requires only basic IMS knowledge
  - Supports all major IMS capabilities

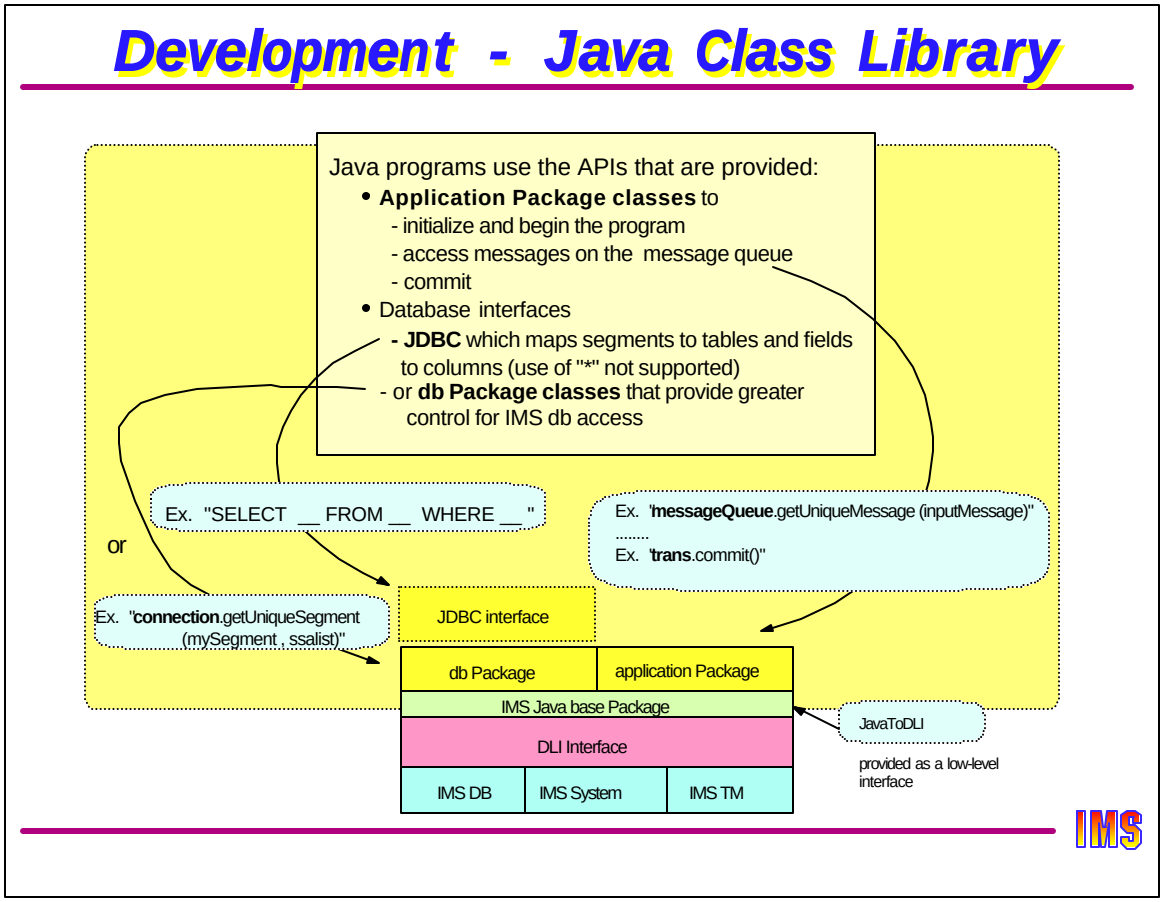
IMS

Most college graduates who have taken programming classes in college have learned Java. Fewer are learning the more traditional languages of COBOL, PL/I or Assembler. For this reason alone, IMS support of application programs written in Java is a huge benefit.

Supports both the Java Development and Java Runtime environments. The development environment uses a set of Java Classes which can be used by Java programmers to access IMS databases, the message queues, and to request IMS services such as commit and rollback processing. These are delivered with IMS V8 as part of the Java Class Library.

The runtime environment includes two new dependent region types which can only be used to execute programs written in Java. These dependent region types, called the JMP (Java Message Processing) region and JBP (Java Batch Processing) region, establish a "Persistent Reusable Java Virtual Machine" within the dependent region address space. This achieves an important performance advantage over the prior IMS support of the HPJ (High Performance Java) compiled program running in an MPP or BMP region by not collapsing and reestablishing the environment with every transaction.

Note that IMS V8 does not support HPJ programs. These programs must be modified to run in the JVM.



IMS Java programmers use APIs delivered with IMS V8.

- Application Package classes are used initialize the program, to access the message queues, and to commit (basically, to perform functions normally associated with the IO-PCB or ALT-PCB).

Database interfaces

- JDBC interface which allows the program to issue SQL-like calls to DL/I databases.
- DB Package classes are used to access the IMS databases in a manner more similar to the standard DL/I interface (GU, GN, REPL, etc.), only using an API more familiar to the Java programmer.

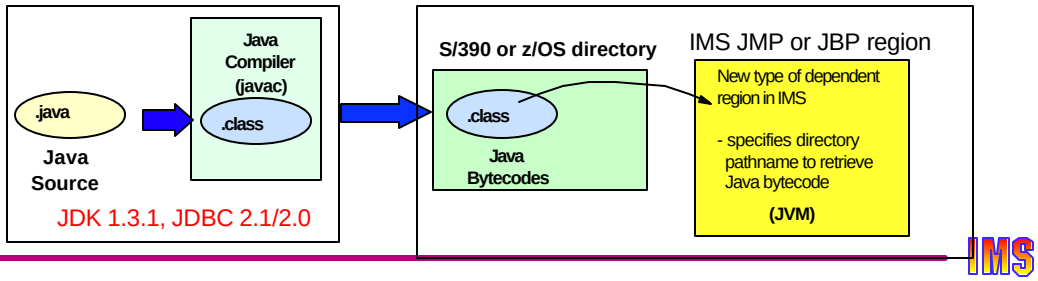
Runtime - New JVM support

Persistent Reusable JVM

- Supports a Java Virtual Machine (JVM) in an IMS environment
  - *Two new dependent region types in IMS*
    - **JMP** for message-driven programs
    - **JBP** for non-message driven programs

Value

- Supports the traditional Java execution model
  - Write the Java program and compile in development environment
  - Send the byte code (.class) to the execution environment



These dependent region types, called the JMP (Java Message Processing) region and JBP (Java Batch Processing) region, establish a "Persistent Reusable Java Virtual Machine" within the dependent region address space. This achieves an important performance advantage over the prior IMS support of the HPJ (High Performance Java) compiled program running in an MPP or BMP region by not collapsing and reestablishing the environment with every transaction.

The application programmer creates the source using the Java Development Kit (JDK), JDBC (if desired), compiles it using the Java compiler, and sends the compiled program (called Byte Code) to the OS/390 or z/OS environment. The program itself is actually a "class." The new dependent regions specify a directory path name to retrieve the "program" and execute it, similar to retrieving a program from STEPLIB except that the program is in a different library.

Note that IMS V8 does not support HPJ programs. These programs must be modified to run in the JVM.

IMS V8 Part Ia Review

Availability and Recovery

- DBRC Enhancements
  - Automatic RECON loss notification
  - RECON command authorization support
  - 16MB RECON record size
  - PRILOG compression
  - Miscellaneous
- Database Image Copy 2 enhancements
- ORS (Online Recovery Service) enhancements
- IMS/DB2 coordinated disaster recovery support
- Dynamic allocation of SLDS after IMS restart
- Batch Resource Recovery Services (RRS) support

IMS

IMS V8 Part Ia Review ...

Performance and Capacity

- Parallel database open processing at restart
- Fast path DEDB enhancements
  - Support for 2048 areas
  - Support for Non-recoverable DEDBs

Systems

- Syntax checker
- Transaction trace

IMS

IMS V8 Part Ia Review ...

Applications

- Java
  - JVM Dependent Regions
  - Java Classes
  - IMS Connector for Java
- APPC
  - Dynamic descriptors
  - Outbound LU support
  - CPU time limit for CPI-C transactions

IMS

Topics Still To Be Discussed

Parallel Sysplex Enhancements (Part Ib)

- IMS exploitation of enhanced XES functionality for structures
  - Autoalter
    - System can alter structure size and entry-to-element ratio
  - System managed rebuild
    - Can move structure without connector being active
  - System managed duplexing
    - System maintains copy of structure
- Shared queues support for synchronous APPC and OTMA transactions
  - Completes the shared queues support for all statndard IMS transaction types

Common Service Layer (Part II)

- Improved systems management of an IMSplex

IMS

These topics will be discussed in the next parts of this set of presentations. They tend to be of interest only to those using, or with plans to use, IMS in a parallel sysplex, or to those whose technical curiosity has got the best of them.

Parallel sysplex enhancements includes enhancements to current usage of the parallel sysplex by IMS applications for data sharing or shared queues.

The Common Service Layer, which also exploits the parallel sysplex, is an architecture which enables a new set of systems management functions entirely different from what has been available in prior IMS parallel sysplex function.

Common Service Layer (CSL)

The next step in IMS architectural evolution

- New *address spaces* built on Base Primitive Environment
  - *Structured Call Interface (SCI)*
    - IMSplex member registration
    - Communications between IMSplex members
  - *Operations Manager (OM)*
    - IMSplex-wide command entry and response
  - *Resource Manager (RM)*
    - Global resource and process management
    - VTAM terminal/user status recovery
- Enables new systems management *functions* in IMSplex
  - *Sysplex Terminal Management (STM)*
    - Uses SCI and RM
  - *Single point of control (SPOC) and user-provided automation (AOP)*
    - Uses SCI and OM
  - *Coordinated Online Change (Global Online Change)*
    - Uses SCI, OM, and RM

IMS

The Common Service Layer is an "architecture" for the IMSplex. In its Version 8 state, it includes three new address spaces which enable enhanced systems management functions. The CSL and the systems management functions will be discussed in detail in a later presentation.

IMS Version 8 Highlights

Enhancement highlights

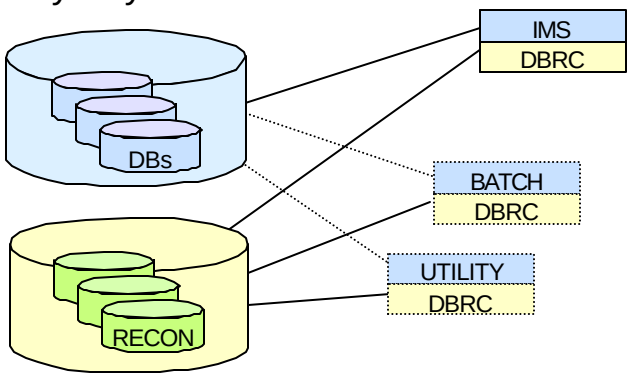
- ★ Base environment
- ★ Parallel sysplex environment
  - Review of resource sharing by IMS through V7
  - Structure management enhancements
  - SQ support for synchronous APPC/OTMA transactions
- ★ Common Service Layer

IMS

In the Beginning

Before IMS/VS V1 R2

- No data sharing
  - Only one IMS at a time could access the data
  - Databases protected by user (DISP=OLD)
- To process database in batch or utility region
  - /DBR database - PROCESS - /START database online
- DBRC used for recovery only
  - No authorization processing



IMS

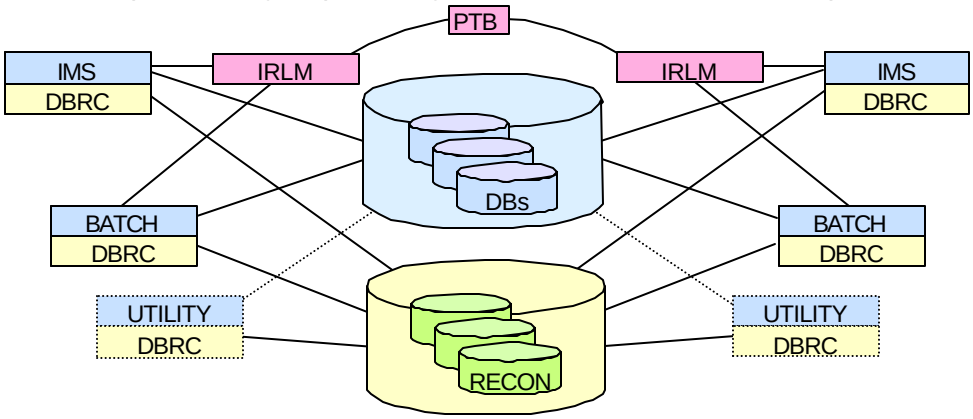
IMS V1 R2 introduced IMS-manged data sharing. Prior to IMS 1.2, data could only be shared at the users peril by coding DISP=SHR on the DBDS DD statement.

To process a database (safely) in another IMS (such as an IMS batch region), it was necessary to DBR the database from the online system, process it in batch, and then restart it in the online region.

Then in IMS/VS V1 R2

Block level data sharing introduced

- DBRC added database authorization processing
- IRLM added as global lock manager (maximum of two IRLMs)
- IMS used IRLM for lock management and buffer invalidation
- IRLMs communicated using Pass-the-Buck processing
  - PTB processing required significant overhead for lock requests



IMS

Version 1.2 brought data sharing in all its glory - well, almost all. DBRC assumed responsibility for database authorization processing, which meant that even with DISP=SHR, if DBRC did not indicate that a registered database could be shared, then authorization processing would fail.

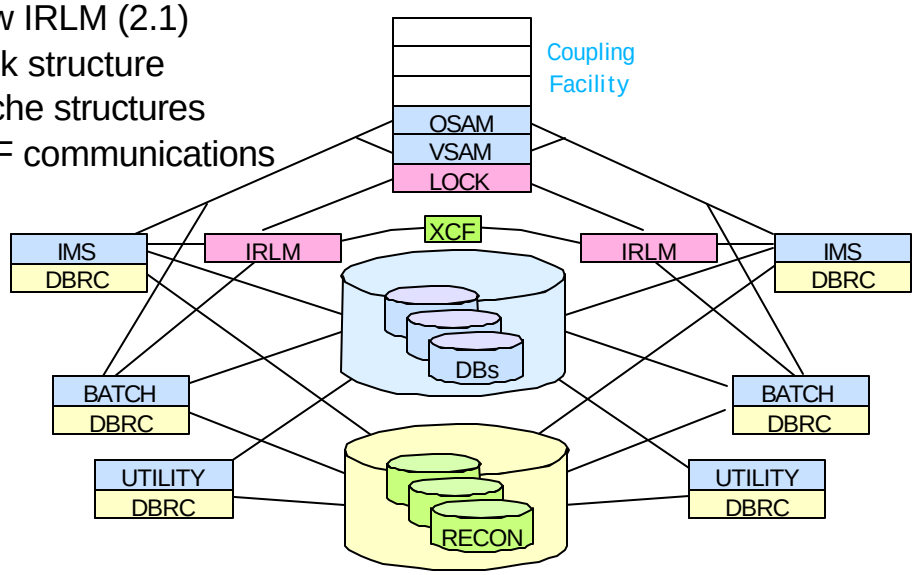
Global lock management became a function of a new address space called the IRLM, which at that time meant IMS Resource Lock Manager. When data was being shared across multiple MVS images, the IRLMs granted locks and coordinated lock management with each other using a process called "Pass-the-Buck (PTB)" processing. This was usually by means of a VTAM link between the two IRLMs, and was, especially by today's standards, but even in those "good ole" days, quite slow.

There was a maximum of two IRLMs allowed in the data sharing group, although one IRLM could support multiple IMSs. Many early IMS data sharing users were between an IMS online system and IMS batch jobs, or between CICS Local DLI and IMS Batch.

After a Long Wait

IMS/ESA V5 started to exploit XCF, the Parallel Sysplex, and coupling facility structures

- New IRLM (2.1)
- Lock structure
- Cache structures
- XCF communications



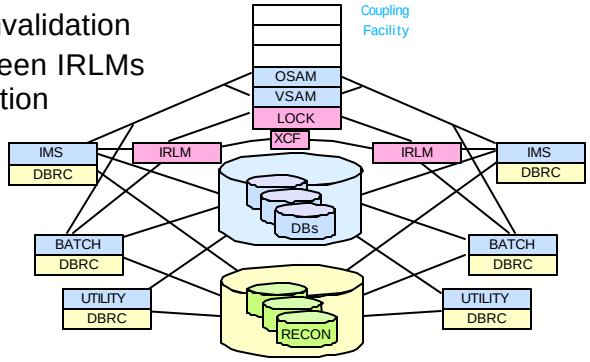
IMS

Not much happened between IMS 1.2 and IMS 5.1. But in V5 the technology changed to take advantage of parallel sysplex functionality. A new IRLM (now called the IBM RLM) was delivered which uses XCF communications services and XES lock services to manage locks. IMS uses XES cache services for buffer coherency (buffer invalidation). In V5, no data was cached in the CF structures.

After a Long Wait ...

Issues addressed in IMS/ESA V5

- Increased capacity
  - More IMSs could participate in data sharing
    - Up to 32 IRLMs
    - Up to 256 IMSs
- Improved performance
  - CF access faster than PTB for lock management and buffer invalidation
  - XCF communication between IRLMs faster than PTB for contention resolution, etc.



IMS

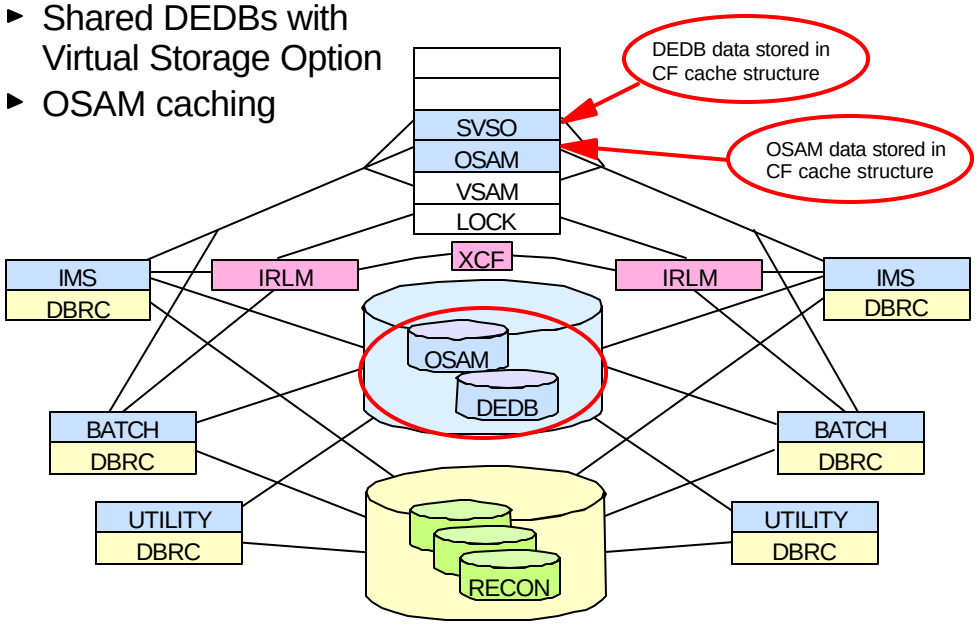
The number of IRLMs in a data sharing group increased from 2 to 32 (the limit on the number of connectors to a lock structure). The number of IMS systems sharing data was 256 (the limit on the number of connectors to a cache structure).

The use of XCF communications, together with the improvements that XES lock services and cache services provided, dramatically improved the performance of IMS data sharing.

Now We're on a Roll

IMS/ESA V6 added additional data sharing support

- Shared DEDBs with Virtual Storage Option
- OSAM caching



IMS

Version 6 added to IMS's exploitation of parallel sysplex data sharing by supporting the caching of IMS data in cache structures.

OSAM used the "store-through" structure to keep copies of data in the structure AFTER it was written to DASD. The intent was to minimize the impact of buffer invalidations, but it also allowed the use of the cache structure as a global buffer pool.

Shared DEDB VSO areas used the "store-in" structure to keep copies of data BEFORE they were written to DASD. At sync point time, committed updates would be written to the structure but not written to DASD until system checkpoint. This greatly improved the performance of DEDB update processing.

Now We're on a Roll ...

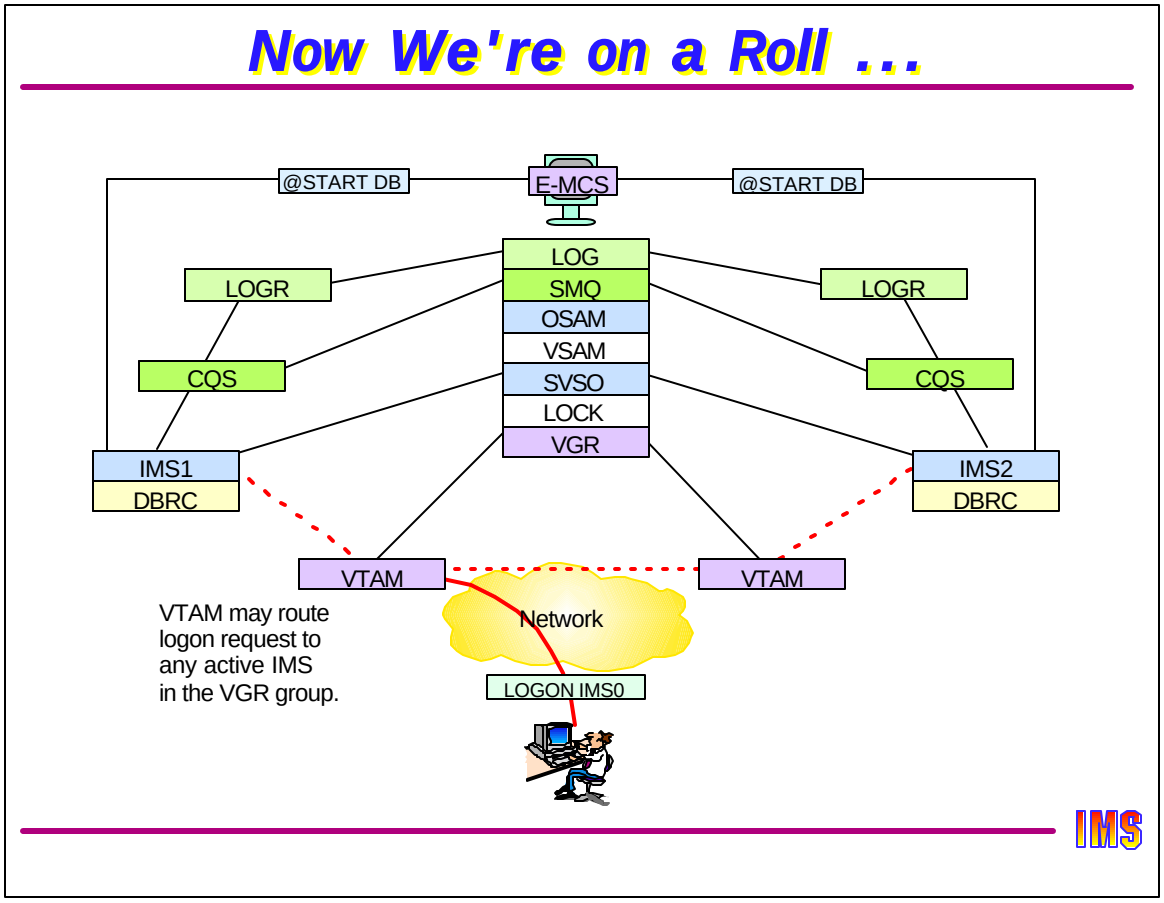
IMS/ESA V6 added additional transaction manager and operations support ...

- Shared message queues for full function and fast path EMH
  - Single set of message queues for all IMSs
    - Queues stored in CF list structures
  - New Common Queue Server (CQS) address space to manage the queue structure
    - CQS uses system logger to log updates to queue structure
- VTAM Generic Resource support
  - Single system image to end user (LOGON IMS)
- Sysplex communications
  - Use of CRC from E-MCS console to send commands to all IMSs in Sysplex and receive responses
- Automatic Restart Manager support
  - Restart IMS following IMS or MVS failure
  - Provided to IMS V5 via PTF

IMS

Also in IMS V6, several new features were introduced.

- Shared Queues allowed the sharing of IMS input and output messages across all IMSs in the "shared queues group." This allowed one IMS to pick up some of the workload of another busy, or failed, IMS. The end user was never aware of which IMS his transaction executed on. Shared queues used a List Structure in the CF, and a server address space called CQS to access the structure, similar (but not the same) to the way the DLISAS address space is used to access databases.
- VTAM Generic Resources was supported which allowed an end-user to log on to a generic name for the IMS data sharing or shared queues group. VTAM would route the logon request to any active IMS.
- The use of a Command Recognition Character (CRC) was supported to allow the routing of commands to any or all IMSs from an MCS or E-MCS console.
- And finally, IMS registered with the Automatic Restart Management function of XCF to enable automatic quick restart of any IMS control region, CQS, or IRLM after abend or system failure.

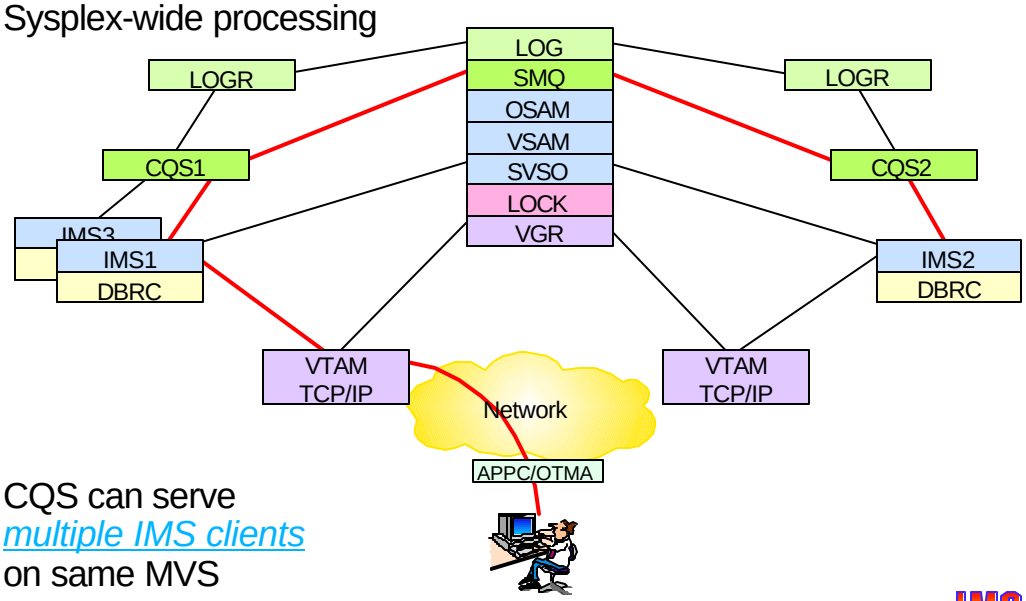


Thsi diagram just shows IMS with CQS, the system logger, the CF structures, and VTAM generic resources. An E-MCS console is shown sending a command to all IMSs with a CRC=@.

IMS V7 Added Still More

Enhanced shared queues support

- *Asynchronous* APPC and OTMA transactions eligible for Sysplex-wide processing



- CQS can serve *multiple IMS clients* on same MVS

IMS

Shared queues lacked support for transactions entered from APPC or OTMA sources. Although the transactions were put on the shared queue, they could only be executed on the front-end IMS (the one which received the input). This, of course, eliminated all of the shared queue benefits for these types of terminals.

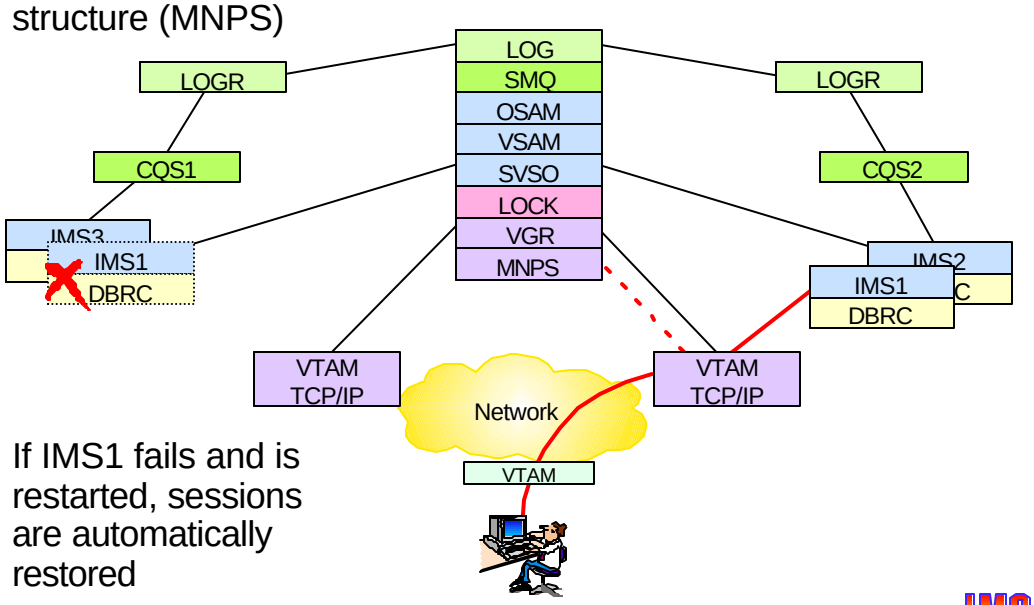
Version 7 added support for "asynchronous" APPC and OTMA input, allowing these transactions to execute on any capable IMS in the shared queues group.

CQS was also enhanced to support multiple IMSs on the same MVS image, as long as they were all in the same shared queues group.

IMS V7 Added Still More ...

Rapid network reconnect

- VTAM session information saved in data space (SNPS) or CF structure (MNPS)



- If IMS1 fails and is restarted, sessions are automatically restored

IMS

IMS V7 also added support for VTAM Multi-node Persistent Sessions (MNPS). IMS calls it Rapid Network Reconnect, but when it is turned on for IMS, VTAM keeps pertinent session information in a CF list structure. If that IMS fails, VTAM will retain this information until IMS is restarted, then those sessions will be reestablished without the user having to log on again and go through the bind process. When thousands of users are logging on following an IMS failure/restart, this could be quite significant.

IMS Version 8 Highlights

Parallel sysplex enhancements

- ★ CF structure management
  - Autoalter
  - System managed rebuild
  - System managed duplexing
- ★ SQ support for synchronous APPC and OTMA transactions

IMS

That brings us to IMS V8. There are several new enhancements for parallel sysplex in IMS V8, including support for some of the more recent XES services for structure management, and well as shared queue support for "synchronous" APPC/OTMA transactions.

Alter and Auto Alter Support

Alter changes a structure without rebuilding it

- Change is made to the existing structure
- Size may be changed by operator command or by connector
- Entry-to-element ratio may be changed only by connector
  - IMS (or CQS) does not change this ratio
- Initiated with operator command:

SETXCF START,ALTER,STRNM=<NAME>,SIZE=<new-size>

Autoalter allows an alter to be done by the system

- Dynamic adjustment
- Size and/or entry-to-element ratio may be changed by system
  - Reacts to needs of the structure
    - Reaching *fullthreshold* causes Auto Alter to increase size
  - Reacts to needs of the system
    - Unused space in structure makes it available for contraction

IMS

ALTER is an XES function which can change the size of a structure within the limits set by the SIZE and MINSIZE parameters in the CFRM policy, or the entry-to-element ratio may be changed. ALTER is done without "rebuilding" the structure. That is, the "alter" is done with the structure remaining in place.

Alter may be initiated by operator command, as shown on the visual, or by a connector, such as CQS. When initiated by command, only the size can be changed. A connector can also alter the entry-to-element ratio, although CQS or any of the other IMS-related connectoars do this.

In prior versions, an active connector was required to perform the alter. For example, when the command is entered to alter the size of a shared queues structure, it is passed to CQS to issue the appropriate XES request. If no connector is available, the command is rejected.

Beginning with IMS V8 (and retro-fitted to V7 through by APAR), a process known as AUTOALTER was supported. This process allows the system to alter either the size or entry-to-element ratio when it becomes necessary. AUTOALTER may be triggered by a shortage of space within the coupling facility, in which case the allocated size of the structure could be decreased, or by the structure reaching the structure FULLTHRESHHOLD as defined in the CFRM policy. For the latter, the size and/or the entry-to-element reatio may be altered. In-use entries or elements will never be taken away, and the size will never be reduced below the MINSIZE specified in the CFRM policy.

Alter and Autoalter ...

Autoalter hardware and software requirements

- OS/390 V2R10; CF Level 9

Autoalter enabling requirements

- Update and activate CFRM policy
  - Administrative Data Utility: IXCMIAPU

ALLOWAUTOALT(YES) - default is NO
FULLTHRESHOLD(percentage) - default is 80%
MINSIZE(nnnn) - default is 75% of INITSIZE

Supported for

- Shared VSO, Shared Queues, IRLM, OSAM, and VSAM
- Resource structure in V8

IMS

Since OS/390 RV2R10 and VF Level 9 are base requirements for IMS V8 with data sharing, no additional requirements are needed for autoalter support.

The CFRM Policy which contains the structure definition must contain the ALLOWAUTOALT(YES) parameter. To invoke autoalter when a structure reaches a percentage full threshold, a FULLTHRESHHOLD value must either be coded or allowed to default to 80%. MINSIZE, although not a requirement, should also be coded to prevent the structure from being contracted below some user-determined reasonable size.

AUTOALTER is supported for all IMS-related structures, although caution should be used for enabling it. The lock structure can be (auto)altered, however, the size of the lock table will not change - only the number of elements for the record list. It is also not recommended for shared VSO structures, especially if they are duplexed or non-preloaded. Perhaps the best use of it would be for the shared queue structures, since autoalter can change the entry-to-element ratio if the structure approaches the full threshold. CQS will not do this. Since autoalter may take away unused entries or elements, using it for OSAM or VSAM may cause these structures to be contracted during periods of low IMS activity, and then be not large enough during periods of high activity. The same is true of the Resource Structure, which may be quite empty during periods of low activity (few logged on users).

System-Managed Rebuild

System-managed rebuild

- Allows operator to move structure to another CF
  - *Does not require an active connector* to rebuild (copy) a structure
  - Operator initiates rebuild with a command
    - If connector active, connector will rebuild,
    - If no connector active, system will rebuild

      SETXCF START,REBUILD,STRNM=<name>,LOC=OTHER

    - Moves one structure to another candidate CF

      SETXCF START,REBUILD,CFNAME=<name>

    - Moves all structures on named CF to other candidate CFs
- Does not provide automatic rebuild of structure for failures
  - This function is provided by system-managed duplexing

IMS

Structure REBUILD is used to move it from one CF to another. If the CFRM Policy is changed prior to the rebuild, then it will assume the properties described in the new policy (for example, to increase the SIZE of the structure.

Prior to V8, rebuild was only supported by an active connector. If no connector is active when the REBUILD command is entered, then the structure will not be rebuilt. This could be a problem if the user wanted to move a structure to another CF while the connectors are down.

IMS V8 (any V7 by APAR) adds system-managed rebuild capabilities to IMS. System-manged rebuild is intended for planned reconfiguration scenarios. The system allocates the new structure, propagates the necessary structure data to the new structure, and then switches over to using the new structure instance.

Note that this does not provide protection against structure of CF failure. This must be provided by system-managed duplexing, or by the connectors themselves.

System-Managed Rebuild ...

Hardware and software requirements

- OS/390 V2R8; CF Level 9

Enabling requirements

- CFRM CDS must be formatted to support SMREBLD
  - CDS Format Utility: IXCL1DSU

```
DATA TYPE(CFRM)
ITEM NAME(SMREBLD) NUMBER(1)
```

- Update and activate CFRM policy
  - Administrative Data Utility: IXCMIAPU

```
PREFLIST(CF01 CF02)
```

Supported for

- Shared VSO, Shared Queues, IRLM
- Resource structure in V8

IMS

As with AUTOALTER, system-managed rebuild is supported by the base hardware and software requirements of Version 8.

The CFRM Couple Data Set (CDS) must be reformatted using the CDS Format Utility to include the SMREBLD functionality. The CFRM Policy may need to be updated to provide a second candidate CF if not already defined.

SM rebuild support is provided for Shared VSO, Shared Queues, IRLM, and the Resource structures.

System-Managed Duplexing

System-managed duplexing

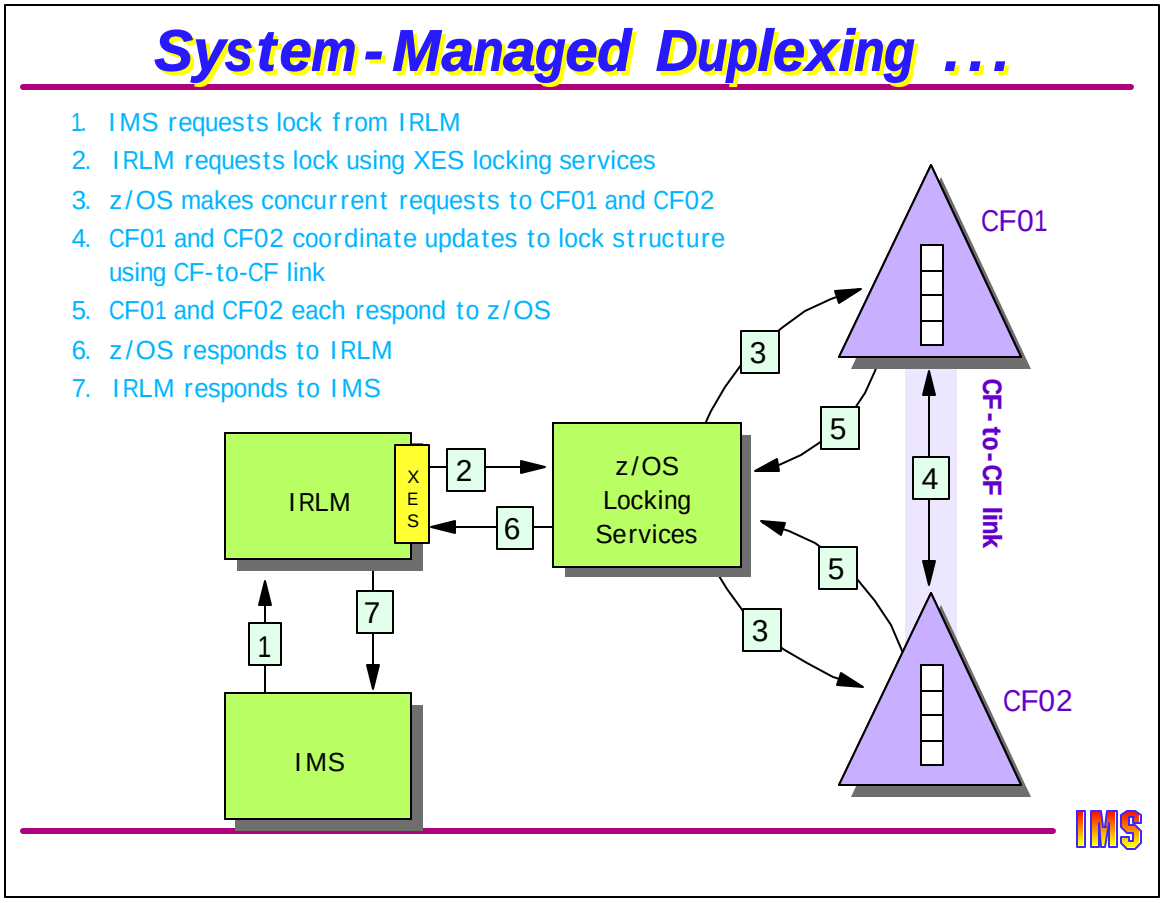
- Objective is to provide
  - Robust failure recovery of a structure with ...
  - Minimal participation by connectors
    - Compare to CQS recovery of shared queues structures
- Provides protection against
  - Structure failure
  - CF failure
  - Loss of CF connectivity
- System creates and maintains two copies of the structure
  - On two different Coupling Facilities
  - One structure name is used for the two structures

IMS

System-managed duplexing is intended to provide a failure recovery capability by using the unaffected structure instance when one of the duplexed pair fails. Failures from which duplexing can recover include structure failures, CF failures, or just plain loss of connectivity. Unlike user-manged duplexing for shared VSO structures, only a single structure is defined in the CFRM policy.

If a CF is available when a failure occurs, the system automatically builds a new structure so that duplexing is maintained. The system performs the significant steps in the duplexing rebuild process, including allocating the secondary (new) instance of the structure, attaching users to the new structure instance, copying all necessary data from the primary (old) instance to the secondary instance, and then transparently duplexing coupling facility operations to both instances of the structure. Since it is system managed, a connector is not necessary to the process.

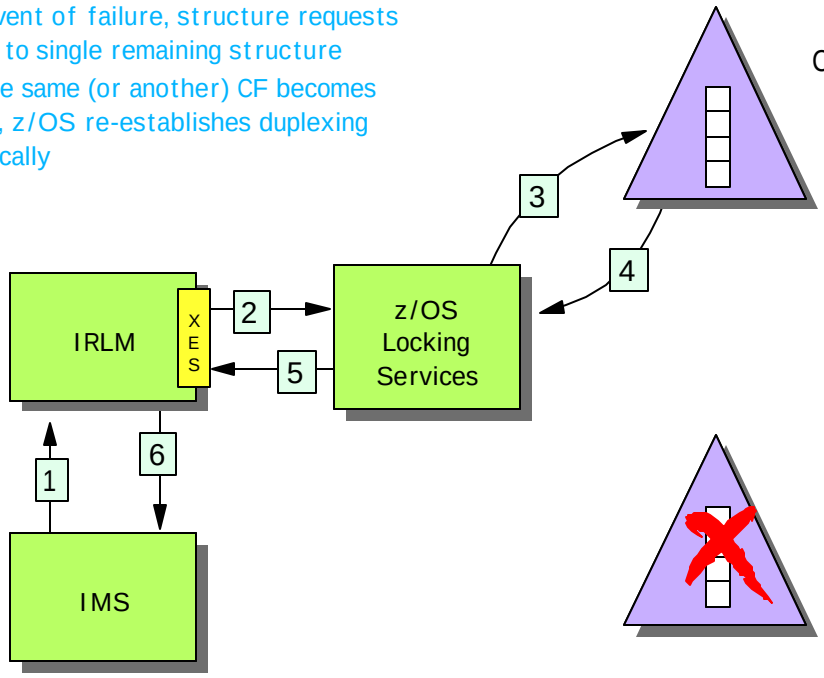
If another CF is not available, operations continue with one structure. If a second CF subsequently becomes available, the system reinstates duplexing by building a new instance of the structure.



This diagram shows some of the major steps in the IRLM using XES locking services and a lock structure to acquire database locks for IMS. The IRLM uses XES locking services to make lock requests. z/OS's XES lock services is shown in the diagram as though it were a separate address space, but it is not - is it z/OS code that is invoked by and executed by the IRLM. Note that this diagram is conceptual and doesn't necessarily include all the steps is acquiring a lock.

System-Managed Duplexing ...

- In the event of failure, structure requests continue to single remaining structure
- When the same (or another) CF becomes available, z/OS re-establishes duplexing automatically



IMS

If (when) a lock structure becomes unavailable to the lock manager, lock services continues with the surviving structure. If another candidate CF were available at the time, another instance of the structure would have been created immediately. If not, then when one does become available, duplexed structures will be reestablished. For a CF to be a candidate, it must be defined in the CFRM policy PREFLIST.

System-Managed Duplexing ...

Enabling requirements:

- CFRM CDS must be initialized with SMDUPLEX specified
  - CDS Format Utility: IXCL1DSU

```
DATA TYPE(CFRM)
ITEM NAME(SMDUPLEX) NUMBER(1)
```

- CFRM policy definition for structure:
  - DUPLEX(ENABLED) - automatically activated
    - or -
    DUPLEX(ALLOWED) - activated by command

```
SETXCF START,DUPLEX,STRNM=name
```

Duplexing supported for

- Shared VSO, Shared Queues, IRLM
- Resource structure in V8

IMS

Before defining duplexed structures, the CFRM couple data set must be formatted for system managed duplexing and for system managed rebuild (the initial duplexing is, in fact, a system managed rebuild).

The CFRM policy has two options
- DUPLEX(ENABLED) means that when the first connector creates the structure, a duplexed copy will be created.
- DUPLEX(ALLOWED) means that at any time after the structure has been created, it may be duplexed by issuing the command SETXCF START,DUPLEX,STRNM=xxxx

Duplexing must also be allowed by the connector. The IMS-related connectors allow duplexing for shared VSO, shared queues, and the IRLM lock structures. In IMS V8, duplexing is allowed for the Resource Structure. Note that it is not allowed for VSAM and OSAM cache structures. This is because the cost outweighs the benefits. There is little impact if one of these fails - IMS merely invalidates all buffers, allocates a new structure, and then starts all over again. A small performance hit at the time, but unlikely to happen and having no integrity issues.

System-Managed Duplexing ...

Hardware and software requirements:

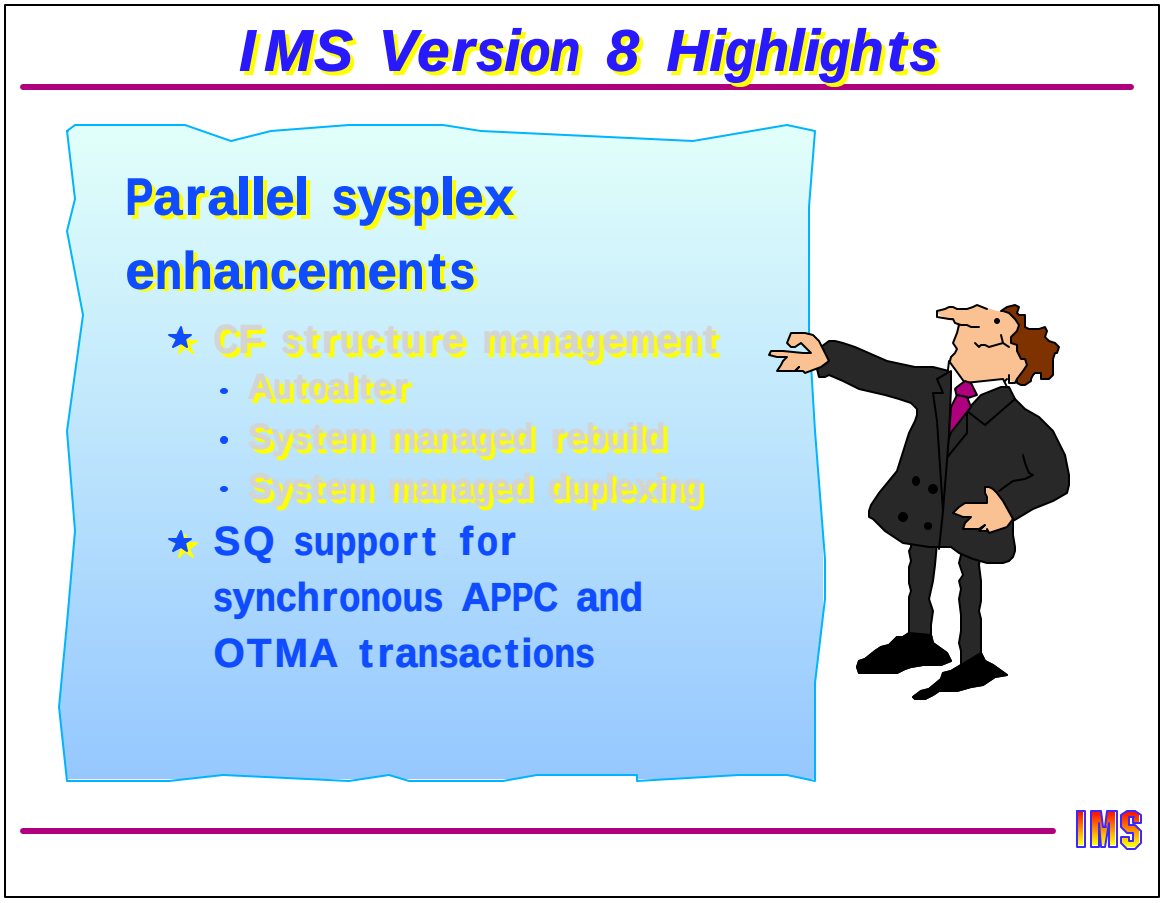
- z/OS V1.2 with APARs OW41617 and OW45976
- CF Level 11 (9672 CF) or CF Level 12 (zSeries CF)
- CF-to-CF links
- IMS V8
- IMS V7 with APAR
  - Call the support center to get the most current list of APARs

IMS

This chart shows the hardware and software requirements for SM duplexing. Note that there are requirements over and above what was required for SM-rebuild and autoalter.

The CF level required depends on the hardware type used for the coupling facility. If the CF is 9672-type hardware, then CF11 is required. If it is zSeries hardware, then CF12 is required. This does NOT refer to the hardware level of the Central Processor. Also, a new type of CF link is required - CF-to-CF links, which are used by the CFCC to coordinate the duplexed writes.

All of the "CF structure" enhancements (autoalter, SM-rebuild, and SM-duplexing) are also available to IMS V7 by APAR. Call the support center to be sure you have the right APARs.



The second category of sysplex enhancements in IMS V8 is support for synchronous APPC and OTMA transactions in a shared queues environment.

Sync APPC/OTMA SQ Support

Description

- Allows synchronous APPC/OTMA transactions to execute on any IMS in the shared queues group
  - APPC Synchronous inbound requests (Allocate - Send - Receive)
  - OTMA Send-then-Commit (Commit Mode 1)

- Automatically enabled if all of the following are true
  - All environments are z/OS V1.2 with RRS enabled
  - Shared queues is enabled
  - All IMS systems in the SQ group are IMS V8
    - Update RECONs to reject V6/V7 signons

```
CHANGE.RECON MINVERS(81)
```

IMS

The new capability allows any IMS in a Shared Queues group to process synchronous APPC and OTMA inbound messages. In prior releases, synchronous messages had to be processed on the IMS system that maintained the connection to the client.

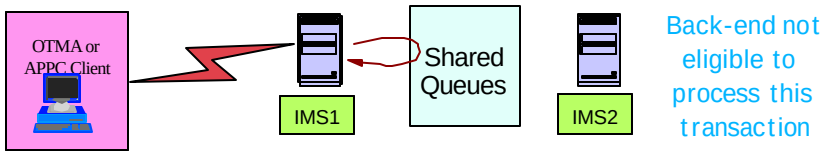
APPC synchronous messages are those sent in by a verb set of Allocate-send-receive. OTMA synchronous messages are those that carry a commit mode of Send-then-Commit. In either case, synchronous processing is characterized by a partner client program sending in a message and waiting for a reply. The IMS that receives the connection request maintains information about the connection and subsequently uses the connection to send the output reply.

There is no parameter or command to enable/disable APPC/OTMA synchronous Shared Queues support. The support is automatically enabled based on a combination of prerequisite conditions. These include: a Shared Queues environment with a minimum level of IMS V8 for all the IMS systems in the group, a DBRC recon specification of "minvers(81)", associated operating system levels at a minimum of z/OS V1.2 along with RRS APAR OW50627, and active RRS (Resource Recovery Services) support on each of the appropriate z/OS systems. The RRS capability that allows the synchronous request to be routed to another system is called "cascaded transaction support".

Background

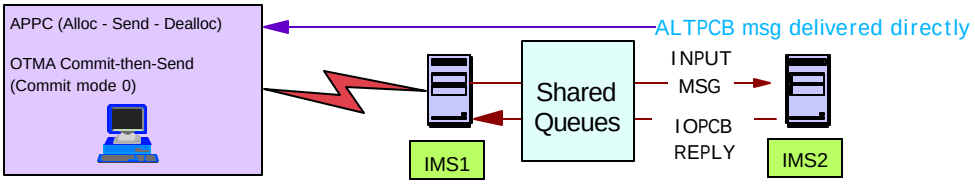
IMS V6 - Introduced Shared Queues support

- ***All*** APPC and OTMA messages processed on SQ "front-end"



IMS V7 - Enhanced Shared Queues support

- ***Asynchronous*** APPC/OTMA messages could process on any system in the Shared Queues group ("front- or back-end")



IMS

When Shared Queues support was introduced in IMS V6, the processing of all APPC and OTMA messages was restricted to the front-end IMS. As shown in the picture, a "front-end" is the IMS to which the APPC or OTMA client is connected and a "back-end" is any other IMS in the Shared Queues group.

With the introduction of IMS V7, the support was enhanced to allow asynchronous messages to be processed by any IMS in the group. This continues to be the way asynchronous messages are processed in IMS V8. For asynchronous support, IMS queues input messages to the appropriate transaction ready queue for processing by any available IMS system. IMS also stores information about the requester (tpipe token or remote LU token) along with the shared message queue (SMQ) name in the input message prefix. This information (token plus SMQ name) is used for output messages and becomes the global shared queue name for IOPCB replies.

If the IMS system that picks up the message for processing happens to be the same system that received the message, then the IOPCB reply is queued to the global shared queue and the appropriate task (OTMA or APPC) is notified that it has output to process.

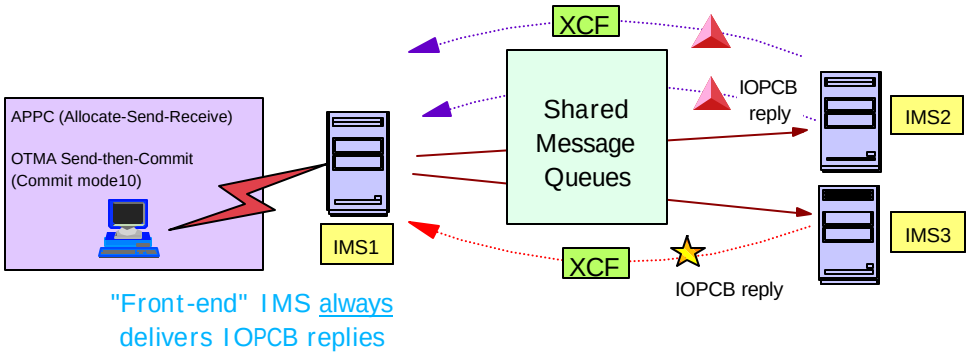
On the other hand, if the message is processed on a back-end system, the front-end is still required to process IOPCB replies. The IOPCB output is queued to the global shared queue but since the front-end system does not register interest in the global shared queue for potential IOPCB replies, a special process is used to notify the front-end system of these messages. This is done by creating a "notify"' message in the back-end that is queued to a specialized OTMA/APPC task in the front-end. Each with a unique queue name. This message is an additional message that is associated with the IOPCB reply. For APPC, the queue name for the "notify" message is "05+DFSAPPCQ+front-end SMQ name". For OTMA, the name is "05+DFSOTMAQ+front-end SMQ name". The front-end registers interest in both these queues. The specialized OTMA/APPC task in the front-end reads the "notify" message which contains the message output queue name, and notifies the appropriate task to process the output.

ALTPCB messages are sent directly from the IMS system that processes the transaction. This system therefore must be able to establish communications with the APPC/OTMA client, meaning that all systems must be APPC and/or OTMA enabled.

Synchronous SQ Support ...

IMS V8 - Completes Shared Queues support

- Synchronous APPC/OTMA message support



"Front-end" IMS always delivers IOPCB replies

Non-conversational IOPCB reply messages (less than 61K) are sent to the front-end using XCF services.

Conversational IOPCB reply messages or any messages greater than 61K are sent to the front-end using Shared Queues along with a special NOTIFY message that is sent using XCF.

IMS

With IMS V8, the restriction on synchronous messages has been removed. When an input message is received from an OTMA/APPC partner, the receiving IMS determines whether the environment is Shared Queues capable and whether the request is asynchronous or synchronous. If it is asynchronous, then the actions described on the previous page are taken. If the message is synchronous then IMS also makes a check to see if the synchronous support is enabled. IMS stores information about the requester (tpipe token or remote LU token) along with the SMQ (Shared Message Queue) name, e.g., IMSID, in the input message prefix.

If the IMS system that picks up the message for processing happens to be the same system that received the message, then the IOPCB reply is not put on the Shared Queues but rather sent directly to the partner client prior to syncpoint processing. This is business-as-usual processing.

If the message is processed on a back-end system then the IOPCB reply must be routed back to the front-end IMS for delivery since it is the front-end that maintains the connection to the client. The routing is done prior to syncpoint processing with the back-end holding the IMS resources until an indication of a commit or abort. Note the following:

- All conversational IOPCB reply messages and messages where all segments added together are greater than 61K, are routed through the Shared Queues with a special "notify" sent via XCF. A specialized OTMA/APPC task in the front-end reads the "notify" message which contains the message output queue name, and notifies the appropriate task to retrieve the message from the Shared Queues.
- All non-conversational IOPCB reply messages less than 61K are sent to the front-end using XCF services.

Synchronous SQ Support ...

Support for ...

- Sync_levels: None, Confirm, Syncpoint
- All transaction types except APPC CPIC-Driven

Uses RRS (Resource Recovery Services) - z/OS V1.2

- Allows synchronization of message-processing and the associated commit across IMS systems

Support is automatically enabled/disabled

- Disabled if RRS not available or not all shared queue members are IMS V8

IMS

It is important to note that the meaning of "synchronous" APPC/OTMA message processing has not changed in this environment. The rules of processing that apply in a non-Shared Queues implementation continue to apply when Shared Queues synchronous processing is enabled. By definition of "synchronous", the partner program continues to wait for a reply message from IMS. This reply can take the form of an IMS application IOPCB reply or a DFS error message. The DFS2082 message which is sent in a non-SQ environment when IMS transaction processing terminates without an IOPCB reply continues to be sent with synchronous SQ enablement. When the front-end IMS processes the input message, the processing is similar to that in non-SQ with the exception that the RRS environment is established. When processing occurs on the back-end, both IMS systems are involved in the RRS commit/backout process. If the back-end IMS system fails then a new message (DFS2224 Back-end system abended) is sent to the remote client.

Input messages can invoke all transaction types except for APPC CPIC-Driven programs. APPC message processed by explicit calls are not passed through the message queues.

APPC/OTMA Shared Queues support is based on RRS services. It is RRS that provides the environment and allows synchronization of message processing across multiple IMS systems.

As mentioned previously, there is no parameter or command to enable/disable the synchronous APPC/OTMA SQ support. Every time an IMS joins the IMSPLEX , all members communicate their current synchronous SQ status (based on adherence to the minimum requirements) to enable/disable the function as necessary. Also, if one of the IMS systems connects or disconnects from RRS, i.e., RRS is enabled/disabled, all members of the group are informed. If all IMS systems and environments meet the requirements, the synchronous support is enabled. If one of the members is unable to support the enablement, it is disabled for the group.

IMS V8 Part Ib Review

Alter and Autoalter support

- Provides improved manageability for VSO and CQS structures

System-Managed rebuild support

- Provides rebuild support for VSO structures
- Provides improved rebuild support of CQS structures

System-Managed duplexing support

- Provides improved availability
  - Eliminates need for /VUNLOAD to reestablish duplexing when a shared VSO structure fails
  - May eliminate need to recover CQS structure
  - May eliminate need to repopulate Resource Structure (new in V8)

IMS

These enhancements will apply to any user of IMS functionality requiring a CF structure. The enhanced function is provided by OS/390 and z/OS through XES. IMS merely allows it.

IMS V8 Part Ib Review ...

Shared queue support for synchronous APPC and OTMA transactions

- Makes the benefits of shared queues available to all types of IMS transactions, regardless of origin
- Requires all IMS systems to be at Version 8 level
  - MINVERS(81) specified in RECONs

IMS

This pretty well completes the IMS terminal/session support for shared queues. The only transaction type not supported now by shared queues is the CPI-C driven transaction, but since IMS itself nevers sees the input message, there is no way it could put it on the shared queue. This support does require that all IMS systems be at V8, AND that the RECONs indicate that the minimum version of IMS that can connect to the RECONs is V8. If MINVERS is set to 71 or 61, even if all connected IMSs are V8, this support will not be enabled.

